QUANTITATIVE ANALYSIS OF ADAPTIVENESS AND CONSISTENCY OF A CLASS OF ONLINE LEARNING ALGORITHMS

A Dissertation
Presented to
The Academic Faculty

By

Carol C. Young

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Robotics

Georgia Institute of Technology

December 2019

QUANTITATIVE ANALYSIS OF ADAPTIVENESS AND CONSISTENCY OF A CLASS OF ONLINE LEARNING ALGORITHMS

Approved by:

Dr. Zhang, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Chernova
College of Computing
Georgia Institute of Technology

Dr. Coogan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Wang
School of Mechanical Engineering
Clemson University

Date Approved: August 15, 2019

I’m excited to see how current and future technologies revolutionize the way we learn.

LeVar Burton

This work would not have been possible without the support of my parents, Sam and Marjorie Young, the encouragement of my sister-by-choice Corinne Livingston, and the unconditional love of Cryus, Prettilena, Archimedes and Apollo.

ACKNOWLEDGEMENTS

The research work is supported by ONR grants N00014-14-1-0635 and N00014-16-1-2667; NSF grants CMMI-1436284 and OCE-1559475; NRL N0017317-1-G001; and NOAA NA16NOS0120028, and an ARCS Scholar Award.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis models online ensemble learning algorithms to obtain theoretical analyses of various performance metrics. Online ensemble learning algorithms often serve to learn unknown, possibly time-varying, probability distributions or interact with other learning systems. Their simplicity allows flexibility in design choices, leading to variations that balance adaptiveness and consistency and allows for chatter resistant co-learning. To analyze online ensemble learning algorithms for these variations this work provides a method for the creation of automata by properly selecting states. These automata provide an analytical framework to quantify the adaptiveness and consistency of online ensemble learning algorithms when interacting with a probability distribution. The resulting Markov chain provides quantatative metrics of adaptiveness and consistency can be calculated through mathematical formulas, other than relying on numerical simulations. This analysis shows that the Multi Expert Algorithm (MEA) achieves a higher consistency than the more adaptive Weighted Majority Algorithm (WMA), and a higher adaptiveness than the more consistent Winnow algorithm, thus achieving a balance between the historical algorithms in terms of the adaptiveness and consistency metrics. The automata also provides an analytical framework to identify chatter which can happen when an online learning algorithm is used by a robot to predict human intention when interacting with a human. When chatter happens, the learning algorithm continually changes its prediction, without reaching a constant prediction of human intention. Utilizing Rescorla-Wagner model for human learning, we analyze an expert based online learning algorithm and identify if chatter will occur, and if so what conditions will cause chatter.

CHAPTER 1
INTRODUCTION

Online ensemble learning algorithms were developed to provide efficient real-time learning. At each time step these algorithms select an output and then receive feedback from the environment about that output. This feedback is then utilized for the selection of the output in the next time step. This allows the algorithm to adapt to an unknown, changing, or learning environment. However for implementation of these algorithms it is important to consider the behavior of the algorithm while learning. If the number of steps to learn is too large, then the benefit to having learning implemented is negligible. If the number of steps to learn is too small then the algorithm can rapidly change outputs, causing failure in implementations that require a consistent output. And if the algorithm is in a co-learning system, that is it is interacting with an environment that is also learning, the system could reach a steady state where the learning done by the algorithm about the environment is counteracted by the learning done by the environment about the algorithm. One example is a robot running an online ensemble learning algorithm while interacting with a human. If the learning time is too large then the robot will not be able to adapt to what the human wants in a time frame that the human will notice. If the learning time is too small the human can perceive the rapid changing of outputs as erratic behavior. And if the human is learning what to expect from the robot, the robot could change outputs when the human updates their expectation, causing the robot to be seen as unpredictable. All of these can cause the human to have a negative perception of the abilities of the robot, leading to a failure of the implementation.

A learning algorithm is considered to be online if it only handles the feedback from each chosen output at the time when the feedback is received. Feedback is not stored. Thus the memory and processing time requirements for an online learning algorithm are minimal

[1].

An online learning algorithm is considered to be an ensemble if it utilizes experts in order to chose the output of the algorithm. Experts are independent methods for choosing an output. Each expert is assigned a weight that determines the confidence that the algorithm has in its output choice. The output with the greatest confidence is then chosen by the algorithm. And then feedback on that output updates the weights of the experts [2]. Ensemble algorithms provide improved performance over individual experts [3]. While experts can be created that are themselves learning algorithms or that produce multiple or overlapping outputs in this paper we will only consider experts that always chose the same unique output.

This class of algorithm has been utilized in various problems, including Human Robot interaction [4], sensorimoter repetitions [5], data fusion [6], pedestrian detection [7], and predicting student performance [8], and for hostile non-stationary environments [9]. The most historically significant examples of online ensemble learning algorithms are the Weighted Majority Algorithm (WMA) and the Winnow Algorithm. Both of these algorithms can undergo a theoretical analysis called error bounds[10], [11]. Error bounds determine the theoretical worst case accuracy of the algorithm when compared to the most accurate expert in the ensemble. For Winnow based online ensemble learning algorithms, there is a non-binary learning algorithm called the Committee algorithm. It assumes that the environment can provide the correct output. This algorithm has been shown to be accurate using error bound analysis and experimental results [12][13]. And while a survey of current online learning literature did not show other theoretical measures for algorithm analysis, experimental accuracy measures was the the most commonly used metric[14].

Some online ensemble learning algorithms have been proposed to better adapt to concept drift. One example is the Dynamic Weighted Majority algorithm that can add and delete experts [15]. Other approaches include regret minimization [16], tracking of the averages [17], and using an age discount factor [18]. These methods were evaluated based on

error rate using numerical experimentation after a drift occurs. However, their adaptiveness has not been evaluated through theoretical analysis.

Relevant to consistency, one consideration is to achieve robustness to noise. Some effort has been reported in the literature. In cases where experts consider multiple parameters, the confidence of each expert over the observed parameters is noted and only experts that exceed a confidence threshold are allowed to vote [19]. In cases where experts are generated in real time from a data stream, an aggregate ensemble method is used to combine multiple frameworks for generating experts [20]. These algorithms have a reduced reaction to noise and less prediction changes. However, the concept of consistency was not explicitly considered, and while there was consideration of error bound, there was no theoretical analysis of robustness or consistency.

However accuracy metrics are not useful for determining the number of steps for learning, nor the behavior when involved with a co-learning system. And previous research has shown that long learning times and rapidly changing outputs are preventing online learning methods from being effectively utilized in human robot interactions [21]. Progress has been made on this problem by employing an online ensemble learning algorithm called Gabe-S++ which separates its experts into groups based on the general strategy each expert employs [22]. The general strategy is then given to the human as feedback, which improved the human's response to the learning algorithm [4]. However this is limited to specific types of implementations and does not address the the ability to determine the number of steps for learning.

Additionally while physical experiments are useful, the theoretical analysis of online ensemble online learning algorithms increases the efficiency of the design process, by allowing algorithm improvements to be analyzed before implementation. In my previous work, we developed two theoretical metrics called consistency and adaptiveness for binary output online ensemble learning algorithms [23]. We define consistency as the expected number of steps that an algorithm will preform the same output in a row. A large value for

consistency means that the algorithm will not rapidly change outputs. We define adaptiveness as the expected number of steps that it takes for an algorithm to learn a new output. A small value for adaptiveness means that the algorithm learns more quickly. This allowed us to determine the number of steps for learning of an unknown and potentially changing distribution by a binary online ensemble learning algorithms. However this method is not sufficient for analysis of a co-learning system.

A co-Learning system is when two learning systems are considered as a single nonlinear system that has its own system dynamics. These co-learning systems are especially important to consider as the amount of learning robots interacting with humans increases. Some human-robot co-learning systems has been been studied using experiments, such as where the robot learns the type of human they are interacting with by using trained examples of humans interacting to select the optimal strategy [24, 25], or by jointly training reinforcement strategies in the framework of a scholastic game so that a robot can use the optimal learned strategy when interacting with a human [26]. However these cases do not consider the potential of the co-learning system to demonstrate *chatter*.

Chatter is a specific limit cycle phenomena where the two learners (human and robot) both adjust their behaviors to correct for an error caused by the mismatch of behaviors. This correction is simultaneous, but both newly learned behaviors are still incompatible, which produce an error that leads to another simultaneous correction. This process repeats indefinitely without settling down on a steady state. This dynamic behavior has been observed in human-human co-learning systems. For example, in pedestrian counterflow studies, when two pedestrians meet, they may block each other repeatedly while trying to pass [27]. There exists evidence [28, 29] showing that human behavior will shift in response to robot behaviors, similar to how human behavior shifts in responses to another human's behavior. Therefore, chatter in human-robot co-learning may exist and should be analyzed.

The first contribution of my thesis is to model non-binary one-vs-all online ensemble

learning algorithm as an automata. The algorithms modeled include the historical algorithms of WMA [10], and Winnow [11], as well as algorithms I developed explicitly to meet the challenges of consistency, adaptiveness, and chatter, called the Multi Expert Algorithm (MEA) and the Human Aware Multi Expert Algorithm (HAMEA). This contribution is in chapter 3. The second contribution is in chapter 4 is to utilize these developed automata with a probability distribution to create a Markov chain model of these algorithms. This Markov chain is then used to calculate values for consistency and adaptiveness. Data collected from human trials is included here. The final contribution is in 5 where the developed automata is utilized to analyze a co-learning system containing an online ensemble learning algorithm and a human learner for chatter. HAMEA is shown to prevent chatter, both theoretically and numerically. This algorithm was also implemented on the GT-MAB [30, 31, 32], showing the platform's feasibility for human robot interaction experiments.

CHAPTER 2

COMPARATIVE ANALYSIS

2.1 Problem Setup

The goal of this chapter is not to create an algorithm that is the most adaptive or the most consistent, but rather an algorithm that can manage the tradeoff between the two. A learning algorithm called the dual expert algorithm (DEA) is introduced to select between two possible control laws, each generate a reaction to one of two choices a human will make. We show that the DEA has bounded errors in selection between two reactions, improving consistency as the number of data points increases, while retaining its ability to adapt to concept drift. We compare the DEA with two similar algorithms (weighted majority algorithm (WMA) [10] and the online Winnow algorithm [33]).

We justify the conclusion and demonstrate real life applications in human robot interaction (HRI) by implementing the algorithm onto a moving robot avoiding a known user approaching it in the hallway. Because pedestrians tend to pass on a predetermined side [34], [35] with parameters that are similar to pedestrians passing other human [36]. The existing methods to solve this problem has the robot stop [37] or treat the humans as obstacles and plan a path around them. [38], [39], [40]. Our method allows the robot to move around a human in a socially aware manor without necessitating large path planing costs.

In the experiment the concept is the internal beliefs of a human that determines which wall they turn towards when encountering a robot moving towards them in the hallway. The two control laws that the robot has to select from are for passing the human by moving towards either the right or the left side of the hallway. The expected output from the robot is that the human and the robot move towards opposite walls to avoid each other. The expected output is then compared to the actual output that is which side of the hallway the

human moves towards. If the two outputs match each other, then the selection made by the learning algorithm is a success, otherwise the selection is an error. The learning algorithm then uses this success or error to improve its next prediction.

2.2 Background

2.2.1 Expert-Based Learning

An expert in an algorithm represents mapping from a set of input to a specific output. In the case of learning binary choices, two experts can be employed: one expert always may the first choice, and the other expert always make the second choice. We can assume that at least one expert will preform as good or better than random guessing. An expert-based learning algorithm in this case will try to select the best expert to believe in. This idea can be generalized to multiple experts. Of all the experts considered there will be one that makes the minimum amount of errors, which we will call the best expert. This best expert is unknown before hand. The learning algorithm will need to discover the best expert by making a number of trials. Errors will be made during these trials, and the algorithm learns from these errors. Thus a feasible learning algorithm must make a bounded number of errors before finding the best expert.

We define *consistency* as how well the algorithm resists different outputs from the same parameters. In the binary case, suppose the first choice is predicted by the best expert, but the concept exhibit the second choice, then a consistent algorithm should not switch its best expert immediately. The switching should only happen when the second choice appears repeatedly. Each time an algorithm changes its prediction, we say a switch occurs. The number of switches is an important parameter for an algorithm used for human robot interaction because it affects the human's perception on the consistency of the robot. A learning algorithm that learns quickly but without consistency will be regarded as unpredictable by human even if the error bound for the algorithm is low.

We define *adaptiveness* as how quickly the algorithm can adapt when the concept that it

is attempting to learn drifts. In the binary case, suppose the concept changes its preference from the first choice to the second choice, then the algorithm should switch its prediction from the first choice to the second choice after a small number errors. In general, for an algorithm to be considered adaptive, it needs to learn the new preference quickly within a minimal number of errors.

2.2.2 Binary Learning Algorithms

The weighted majority algorithm (WMA) [10] and the Winnow algorithm [11][33] are the two basic expert-based learning algorithms that can be used to make selections between two control laws. The WMA using two experts is shown in Algorithm 1. Each expert takes in the parameters of the system and uses them to make their selection. Each expert is assigned a weight that is adjusted by the algorithm. The weights of the experts are then compared, and the expert with the highest weight is chosen. More specifically, the algorithm computes two positive weights $\tilde{W}_1$ and $\tilde{W}_2$ that gauge the likelihood of which choice will be correct in each iteration. A selection is made based on comparing the two weights. If $\tilde{W}_1 > \tilde{W}_2$, then the algorithm chooses expert 1, and if $\tilde{W}_1 < \tilde{W}_2$, then it chooses expert 2. If $\tilde{W}_1 = \tilde{W}_2$, a selection is made randomly with equal probability. If there is an error in the prediction, then the weight correspond to the expert is reduced by half. The Winnow algorithm is very similar to the WMA but has a deviation in line 9 where a correct choice also results in the weight of the corresponding expert being increased by a factor of two.

Both WMA and Winnow have bounded error [10], [11]. And WMA is able to adapt easily to drift, however it is not consistent, since one deviation is enough to cause it to switch its selection. The additional doubling of weights allows Winnow to be consistent but at the cost of its adaptiveness, since now a switch in selection can only be caused when there are an equal number of deviations as there are of the preferred output.

By extending these algorithms, we would like to design an algorithm that is more consistent than the WMA, and more adaptive than the Winnow. It has bounded error like both

Algorithm 1 Weighted Majority Algorithm (using two "experts" with opposite selections)

1: Set $\tilde{W}_1 = \tilde{W}_2 = 0.5$
2: Choose selection a from $\text{argmax}\left(\tilde{W}_1, \tilde{W}_2\right)$
3: **if** $\tilde{W}_1 = \tilde{W}_2$ **then**
4: Choose a randomly from $\{1, 2\}$ with equal probability
5: **end if**
6: **if** Error **then**
7: $\tilde{W}_c = \frac{\tilde{W}_c}{2}$
8: **else if** (Winnow) **then**
9: $\tilde{W}_c = 2\tilde{W}_c$
10: **end if**

the WMA and Winnow while maintaining a balance between consistency and adaptiveness. The Dual Expert Algorithm (DEA) is shown in Algorithm 2. The setup is similar to WMA with the increasing weights of Winnow, however there is one key difference in line 12. When a correct selection is made and the weight is over .25, line 12 increases the weight by taking its square root, which keeps the value $W_a \leq 1$. This bound is the key difference that allows for the DEA to maintain a balance between consistency and adaptiveness.

Algorithm 2 Dual Expert Algorithm

1: Set $W_1 = W_2 = 0.5$
2: Choose selection a from $\text{argmax}\,(W_1, W_2)$
3: **if** $W_1 = W_2$ **then**
4: Choose a randomly from $\{1, 2\}$ with equal probability
5: **end if**
6: **if** Error **then**
7: $W_a = \frac{W_a}{2}$
8: **else if** Correct **then**
9: **if** $W_a \leq 0.25$ **then**
10: $W_a = 2W_a$
11: **else**
12: $W_a = \sqrt{W_a}$
13: **end if**
14: **end if**

2.3 Analysis

Suppose the DEA algorithm has learned the preference of C. Let δ be the number of iterations when C disobeys its preferred option and makes a "deviation". Let the total number of iterations be N, we require that $\delta < \frac{N}{2} - 1$.

2.3.1 Performance

If the concept makes a deviation from its preference, then it will trigger the learning algorithm to make a "wrong" selection based on its past knowledge. The following claim can be made about the error in selection.

Proposition 2.3.1. *The maximum number of selection errors that Algorithm 2 generates is* $1 + 2\delta$.

Proof. Let W_{p} be the weight of the preferred option of the concept C, and let W_{d} be the weight of the deviation option. When $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} > 1$ the preferred selection is chosen. When $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} < 1$ the deviation is chosen. And when $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} = 1$, which is the starting condition, a random choice is made.

When $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} \geq 1$ a deviation can cause an error, leading to a total of δ possible errors. Each deviation can decrease the ratio $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}}$ by at most a factor of 2. Since the ratio starts at 1 the minimum that the ratio can ever be is $2^{-\delta}$. Thus to return this ratio back to 1 it must be increased by a factor of 2^{δ}. This requires there are δ times that the preferred selection is chosen by C while $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} < 1$. Hence the algorithm can make at most δ additional selection errors when $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} < 1$.

Therefore after considering all deviations, $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} \geq 1$. And since the deviation can be chosen when $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} = 1$, but this error will then raise $\frac{W_{\mathrm{p}}}{W_{\mathrm{d}}} > 1$, there can be at most 1 additional error. Thus the total number of errors is less than $\delta + \delta + 1$. Making the error bound of algorithm 2 as $E = 1 + 2\delta$. □

The error bound $1 + 2\delta$ is on the same order of magnitude as the Weighted Majority Algorithm which has an error bound of $E = 2.4 + 2.4\delta$ [10].

2.3.2 Consistency

We show that the selection of the DEA is consistent to deviations made by C. This means that the number of switches between selections of the algorithm are kept small, by ignoring deviations. We say the DEA becomes more consistent as the number of iterations increases the ability of the algorithm to ignore deviations increases.

First we show that for the weighted majority algorithm, the two weights will always be within a factor of two from each other.

Proposition 2.3.2. *Let $\tilde{W}_1$ and $\tilde{W}_2$ be the two weights in the weighted majority algorithm, then $\max\{\frac{\tilde{W}_1}{\tilde{W}_2}, \frac{\tilde{W}_2}{\tilde{W}_1}\} = 1$ or 2.*

Proof. The weighted majority algorithm starts with $\tilde{W}_1 = \tilde{W}_2$. So that initially $\max\{\frac{\tilde{W}_1}{\tilde{W}_2}, \frac{\tilde{W}_2}{\tilde{W}_1}\} = 1$. At any time, when $\frac{\tilde{W}_1}{\tilde{W}_2} = \frac{\tilde{W}_2}{\tilde{W}_1} = 1$ a random choice between the selections is made. And if $\frac{\tilde{W}_1}{\tilde{W}_2} > 1$ then selection 1 is chosen, likewise if $\frac{\tilde{W}_2}{\tilde{W}_1} > 1$ then selection 2 is chosen.

If an error happens in the selection when $\frac{\tilde{W}_1}{\tilde{W}_2} = \frac{\tilde{W}_2}{\tilde{W}_1} = 1$, $\max\{\frac{\tilde{W}_1}{\tilde{W}_2}, \frac{\tilde{W}_2}{\tilde{W}_1}\}$ will be increased to 2. When $\max\{\frac{\tilde{W}_1}{\tilde{W}_2}, \frac{\tilde{W}_1}{\tilde{W}_2}\} = 2$, then if an error is made then since the largest weight is reduced by half. the ratio $\frac{\tilde{W}_1}{\tilde{W}_2}$ will be reduced to 1. □

If C makes a deviation, then the weighted majority algorithm will change the weights to be equal because the larger weight will be reduced by half. The maximum number of switchings an algorithm can make is 2δ where δ is the number of deviations C makes. No deviation is ignored.

Next we will argue that using the dual expert algorithm, the algorithm will make less switchings in its selection because it can ignore deviations made by C.

Let us define $R = \max\{\frac{W_1}{W_2}, \frac{W_2}{W_1}\}$. First we argue that in a number of situations, R will be increased to a value that is greater than 2.

Claim 2.3.3. *Starting from $W_1 = W_2 = 0.5$, if the DEA makes one error followed by one correct selection, then $R > 2$.*

Proof. The weight that is updated is always the larger weight, since a weight must be chosen to be modified, and a weight must be larger to be chosen. This means that when there is an error the maximum weight will decrease by a factor of two, in this case making $\min(W_1, W_2) = .25$. Then the $\max(W_1, W_2) = .5$.

If the next selection is correct then according to algorithm 2 $\max(W_1, W_2) = \sqrt{(.5)} > .5$, which leads to $R = \sqrt{(.5)}/.25 > 2$.

□

Claim 2.3.4. *Starting from $\max\{W_1, W_2\} < 0.25$ and $R \leq 2$, if DEA makes two consecutive correct selections, then $R > 2$.*

Proof. R is the maximum of two positive reciprocal values, thus $R \geq 1$. A single correct selection when $\max\{W_1, W_2\} \leq 0.25$ increases the correct weight by a factor of 2, meaning that R is increased by a factor of 2. Thus $R \geq 2$, and $\max\{W_1, W_2\} < 0.5$. An additional correct selection will then increase the $\max\{W_1, W_2\}$, thus increasing R so that $R > 2$.

□

Claim 2.3.5. *Starting from $\max\{W_1, W_2\} < 0.5$ and $R \leq 2$, if DEA makes one error followed by one correct selection, then $R > 2$.*

Proof. A single error would decrease the max weight by a factor of 2 so that $\min\{W_1, W_2\} \leq 0.25$ and $1 < R \leq 2$.

If $\max\{W_1, W_2\} < 0.25$ then a single correct selection would increase the max weight by a factor of 2 so that $2 < R \leq 4$.

And if $\max\{W_1, W_2\} \geq 0.25$ then a single correct selection would increase the max weight such that $\max\{W_1, W_2\} \geq 0.5$. Then $R = \max\{W_1, W_2\}/\min\{W_1, W_2\} > 0.5/.25 = 2$.

□

Claim 2.3.6. *By making consecutive correct selections, the value of R can be increased until $R = \frac{1}{\min\{W_1, W_2\}}$.*

Proof. Each correct selection only affects the maximum weight. And while $\max\{W_1, W_2\} < .25$, $\max\{W_1, W_2\}$ increases by a factor of 2. Once $\max\{W_1, W_2\} \geq .25$, $\max\{W_1, W_2\}$ is square rooted to form the new maximum. Thus $\max\{W_1, W_2\}$ converges to 1 as the number of consecutive correct selections increase. Since $R = \max\{W_1, W_2\}/\min\{W_1, W_2\}$, and the minimum weight is unchanged, R converges to $\frac{1}{\min\{W_1, W_2\}}$.

□

The number of deviations the dual expert algorithm can ignore depends on the actual ratio between the weights.

Claim 2.3.7. *If $R > 2^K$ where $K > 1$, then $(K-1)$ consecutive deviations can be ignored.*

Proof. Each deviation reduces the ratio R by a factor of 2. Because the larger weight is decreased by a factor of 2. Since $R > 2^K$, the larger weight can be decreased by 2 for a total of $K - 1$ times, and still be the larger weight, thus allowing the algorithm to ignore $K - 1$ deviations.

□

Therefore, if C makes deviations, the DEA will be more robust than the WMA. Since the weighted majority algorithm does not ignore deviations and the DEA does.

2.3.3 Adaptiveness

If we consider drift to be the change in preferred output of the concept C from p to $\tilde{p}$. Then the adaptiveness of the DEA can be shown to be directly linked to the amount of deviations, that have been encountered before drift occurred.

Claim 2.3.8. *R is bounded by the number of consecutive deviations.*

Proof. Claim 2.3.6 showed that the maximum of R is $R = \frac{1}{\min\{W_1, W_2\}}$.

Only an error can decrease weights in DEA and Proposition 2.3.1 showed that the error is bounded by the number of deviations. Thus $\min\{W_1, W_2\}$ is bounded by the number of consecutive deviations.

Therefore $R = \frac{1}{\min\{W_1, W_2\}}$ is bounded by the number of consecutive deviations.

□

Claim 2.3.9. *For any given $2^{K-1} \leq R < 2^K$ after output $\tilde{p}$ is observed for K consecutive iterations, DEA will change from its selection p to the alternate selection $\tilde{p}$.*

Proof. Each time the output is $\tilde{p}$ and DEA selects p there is an error. This error decreases the maximum weight, and thus R is reduced by a factor of 2.

Then after $K - 1$ consecutive outputs of $\tilde{p}$, $1 \leq R < 2$

If $1 \leq R < 2$ than an addition output of $\tilde{p}$ will decrease the weight corresponding to p to be less than the weight corresponding to $\tilde{p}$. This will change the selection from p to $\tilde{p}$.

Thus a maximum of K consecutive C outputs of $\tilde{p}$ are needed to change the selection of the DEA.

□

These two claims imply that the number of errors needed to change selections is limited by the number of deviations leading up to the change.

This can be compared to Winnow algorithm where the 1 in $R = \frac{1}{\min\{W_1, W_2\}}$ is replaced by $\max\{W_1, W_2\}$, which can be very large. This allows Winnow to be consistent but not adaptive.

2.3.4 Tie Breaking

In DEA as well as WMA and Winnow, it can be easily seen that the algorithm cannot issue a prediction when the the weights of the experts are tied. A tie will force the algorithm to

randomly choosing an expert, which is not preferred. In the following, we show that the DEA has reduced number of ties comparing to WMA or Winnow.

Proposition 2.3.10. *If a tie is created by DEA, then $\delta = \frac{N}{2}$ must be satisfied.*

Proof. Define η as the number of predictions in the preferred choice such that $\eta + \delta = N$. Define r_η as the ratio between the weight of the expert making the preferred choice and the weight of the expert making the other choice. And define $r_\delta = 1/r_\eta$, so that $R = \max\{r_\eta, r_\delta\}$.

When the $\max\{W_1, W_2\} < 0.25$ each δ decreases r_η by a factor of two because each δ will either decrease the weight of the expert making the preferred choice or increase the weight of the expert making the other choice. Likewise each η increases r_η by a factor of two. Therefore when $\eta = \delta$ the ratio between the maximum and minimum weights are identical.

To raise $\max\{W_1, W_2\} \geq 0.25$ a correct prediction is needed. When $\max\{W_1, W_2\} \geq 0.25$, each correct prediction square roots $\max\{W_1, W_2\}$, so that $0.5 \leq \max\{W_1, W_2\} < 1$ for any finite number of correct predictions. And because the weights are always positive only the positive square root result is considered, this means that for each number between 0.25 and 1 that is square rooted there will be a unique solution.

An incorrect prediction can either lower $\max\{W_1, W_2\} < 0.25$ or make the weight a unique number based off of the prior $\max\{W_1, W_2\} \geq 0.5$ which is based off of the number of correct predictions.

Because the maximum weight is the only weight that is ever changed, the minimum weight can only decrease. This means that if one weight goes below 0.25 then all potential ties can only take place when the weight is below 0.25. Thus for a tie to occur each weight would have to be lowered below 0.25, one by an error due to η the other by an error due to δ. Thus $\eta = \delta$

Each time the weight increases above 0.25 due to a correct prediction an incorrect prediction would be needed to lower it again below 0.25. Thus $\eta = \delta$.

Finally because of the weight becoming a unique number based off of the number of the correct predictions while the max weight is above 0.25 and incorrect predictions when the weight is above 0.5 there must be the same number of correct and incorrect predictions for both weights in order for them to be equal. Thus $\eta = \delta$.

This means that for tied weights, $\eta = \delta$ while $\max\{W_1, W_2\} \geq 0.25$ and the ratio must remain the same when below 0.25 which implies $\eta = \delta$ while $\max\{W_1, W_2\} < 0.25$.

Therefore $\eta = \delta$, which is equivalent to $\delta = \frac{N}{2}$, must be true for there to be a tie.

□

Note that even if $\delta = \frac{N}{2}$, there may not be a tie. For example starting at $W_1 = W_2 = 0.5$ a correct prediction on the preferred direction will raise $\max\{W_1, W_2\} = \sqrt{.5}$, then an incorrect prediction, caused by a single δ would lower that weight to $\sqrt{.5}/2 \approx 0.3536$. While leaving the other weight unchanged at 0.5. Since this example uses $\delta = 1$ and $N = 2$, and $0.5 \neq 0.3536$. This serves as an example to show that $\delta = \frac{N}{2}$ may not lead to a tie.

Next we note that the weighted majority algorithm will create a tie whenever one correct prediction is followed by an incorrect prediction, regardless of the relationship between δ and N. In addition the Winnow algorithm will create a tie whenever $\delta = \frac{N}{2}$. Therefore we can say that DEA has a lower chance of generating a tie compared to the weighted majority algorithm and the Winnow algorithm.

2.4 Expanded Dual Expert Algorithm

The DEA uses two weights to select potential outcomes. However an algorithm can encounter situations where the preference depends on a parameter. If the parameter is smaller than a threshold then output 1 is preferred, and if the parameter is larger than a threshold then output 2 is preferred.

An example of this type of parameter would be the relative position that a human starts in a hallway. Starting along the left wall would imply that the human would tend to stay

along the left wall to pass, and starting along the right wall would imply that the human would tend to stay along the right wall to pass. However somewhere between the walls the preference of the wall to pass along changes. We propose the extended dual expert algorithm (EDEA) to learn these preferences using weights that are function of the parameter.

The parameter is discretized into M values. We call each values of the parameter a section and each section is indexed by n. Let us assume that there are two edges $n = 1$ and $n = M$, and that output 1 is the preference of edge $n = 1$, while output 2 is the preference at edge $n = M$. And the two weights are now represented as two M dimensional vectors W_1^i and W_2^i indexed by $i = 1, 2, ..., M$.

The EDEA algorithm in Algorithm 3 updates its weights based on whether a correct or an incorrect selection is made. Line 8 shows that when an incorrect selection is made the weights between n and the edge that prefers the observed output are decreased. And when a correct selection is made the weights between n and the edge that prefers the observed output are increased, as shown in line 10

Algorithm 3 Expanded Dual Expert Algorithm

1: Set $W_1^i = W_2^i = 0.5 \; \forall i = 1 : M$
2: The parameter is in section n
3: Choose selection a from argmax(W_1^n, W_2^n)
4: **if** $W_1^n = W_2^n$ **then**
5: Choose selection randomly
6: **end if**
7: **if** Error **then**
8: $W_a^i = {}^{W_a^i}/_2, \; \forall i = \begin{cases} n : M & \text{if } a = 1 \\ 1 : n & \text{if } a = 2 \end{cases}$
9: **else if** Correct **then**
10: $W_a^i = \begin{cases} 2W_a^i & \text{if } W_a^i < 0.25 \\ \sqrt{W_a^i} & \text{if } W_a^i \geq 0.25 \end{cases} \quad \forall i = \begin{cases} 1 : n & \text{if } a = 1 \\ n : M & \text{if } a = 2 \end{cases}$
11: **end if**

2.4.1 Performance Analysis

In order to discuss the performance of the algorithm we will make the following additional assumptions. We assume that there is a section index m so if $n \leq m$, then the preferred output is 1, and if $n > m$, then the preferred output is 2.

Thus consider that the preference is dependent on the section. Since "deviations" are defined as when the output varies from the concept preference, we must allow deviations to be dependent on sections.

Define δ_n as the number of deviations in section n. Thus the total number of deviations denoted as Δ is $\Delta = \sum_{n=1}^{M} \delta_n$. Under this assumption, the error bound of the EDEA is given below:

Proposition 2.4.1. *The upper bound on the total error for the EDEA algorithm is* $E = (M+1)\Delta + M$.

Proof. There are two potential sources of errors in the EDEA algorithm. Consider section n. An error that occurs when a deviation occurs will decrease the ratio of $\frac{W_{\mathrm{p}}^n}{W_{\mathrm{d}}^n}$ by a factor of 2 in the section where the error took place. Each error that occurs without a deviation will increase the ratio of $\frac{W_{\mathrm{p}}^n}{W_{\mathrm{d}}^n}$ by a factor of 2 in the section where the error took place. Each error that occurs, regardless of source, can affect the ratio in a maximum of M sections and a minimum of 1 section.

The maximum number of errors that can be caused by deviations is the total number of deviations Δ. Since each error due to a deviation can effect a maximum of M sections in the worst case, the ratio of $\frac{W_{\mathrm{p}}^i}{W_{\mathrm{d}}^i}$ for $i = 1...M$ would decrease by a factor of 2^{Δ} . In order to increase this ratio to be greater than 1, the ratio in each section must have an increase $\Delta + 1$ times.

In the worst case, each of these increases would be caused by an error made without a deviation. And because the minimum number of sections that an error can affect is 1, the worst case is that the ratio in each section is brought to be greater than 1 individually. Since

there are M sections this means that there are at most $M\Delta + M$ errors that occur without a deviation. Therefore the bound on the error is $E = (M + 1)\Delta + M$.

□

This error bound can be tightened if the section m where the preference changes is known.

Proposition 2.4.2. *With a known m the error bound for the EDEA algorithm is $E = (\max\{m, M - m\} + 1)\Delta + M$.*

Proof. The error bound given in the previous proof, $(M + 1)\Delta + M$, assumes that one deviation can decrease the ratio of the preferred weight over the deviation weight in M sections. However the deviation is dependent on the section. So a deviation can only decrease the weight of the preferred output in $\max\{m, M - m\}$ sections. This means that the error bound of algorithm 3 can be tightened to be $(\max\{m, M - m\} + 1)\Delta + M$. □

2.4.2 Consistency and Adaptiveness

In EDEA, because weights within sections are updated in the same way as weights are updated in DEA, the ratio of weights in a section determines how many deviations in that section can be ignored.

The consistency also has a dependence related to parameter. Suppose that the same selection is preferred in sections w and s.

Proposition 2.4.3. *If $|m - w| > |m - s|$ then $\frac{W_{\mathrm{p}}^w}{W_{\mathrm{d}}^w} \geq \frac{W_{\mathrm{p}}^s}{W_{\mathrm{d}}^s}$.*

Proof. From EDEA, if W_{p}^s is increased, then W_{p}^w is also increased. And if W_{d}^s is decreased then W_{d}^w is also decreased. But an increase of W_{p}^w does not guarantee an increase of W_{p}^s and a decrease of W_{d}^w does not guarantee a decrease of W_{d}^s. Thus $W_{\mathrm{p}}^w/W_{\mathrm{d}}^w$ increases if $W_{\mathrm{p}}^s/W_{\mathrm{d}}^s$ increases, but is not guaranteed to decrease if $W_{\mathrm{p}}^s/W_{\mathrm{d}}^s$ decreases.

Since $W_{\mathrm{p}}/W_{\mathrm{d}} = 1$ in all sections before the first iteration. It is true that $\frac{W_{\mathrm{p}}^w}{W_{\mathrm{d}}^w} \geq \frac{W_{\mathrm{p}}^s}{W_{\mathrm{d}}^s}$.

□

Because the increase of weights within each section is bounded similar to the bound in DEA, there will be the same bound on the number of errors needed to change selections. Proposition 2.4.3 showed that if $|m-w| > |m-s|$ then $\frac{W_{\mathrm{p}}^{w}}{W_{\mathrm{d}}^{w}} \geq \frac{W_{\mathrm{p}}^{s}}{W_{\mathrm{d}}^{s}}$. This also implies that adaptiveness has a dependence related to the measured variable, and sections near m will be more adaptive than sections near $n = 1$ or $n = M$.

2.5 Simulation

2.5.1 Dual Expert Algorithm

Setup

The learning of a concept C by the DEA, WMA, and Winnow algorithms was simulated. C was created to chose one of two outputs in each iteration of a trial. This output was randomly generated using a constant probability of choosing the deviation. After 50% of the iterations in a trial, drift was added by changing which output was the deviation while keeping the probability of making a deviation constant.

Results

Figure 2.1 shows the average total rate of errors by DEA, WMA and Winnow. DEA always kept the rate of errors smaller than the WMA. Before drift occurred Winnow had a slightly lower error rate than DEA, however after drift occurred, Winnow's error rate increased dramatically while WMA and DEA's remained relatively consistent. This shows that DEA offers error rate comparable to Winnow, while maintaining a drift resistance comparable to WMA.

Figure 2.2 shows that the rate of switches for DEA is lower than WMA and close to Winnow, except for a spike that happens when drift occurs. In addition the number of switches in all cases for DEA and Winnow decrease towards 0, both before and after drift occurs, while the number of switches in the WMA remains relatively constant, as the

number of trials increase.

This shows that DEA manages the balance between consistency and adaptiveness better than the two expert WMA and Winnow.

2.5.2 Expanded Dual Expert Algorithm

Setup

The learning of a concept C by 5 different algorithm was simulated. At the start of the trial the section m, where the preference changes, was selected. Each iteration could take place in one of 10 sections, randomly selected with equal probability. And the output of C was randomly chosen using the preference of the selected section and a constant probability of deviation. After half of the iterations drift was simulated by changing m.

Here is the list of algorithms used to learn concept C.

1. Expanded dual experts algorithm
2. The weighted majority algorithm
3. The winnow algorithm
4. DEA applied to each section individually
5. DEA applied with no regards to sections
6. Multi expert, dual experts algorithm modified to use more than two experts

Both the Winnow and WMA are created according to their traditional multi-expert use [11], [33]. And the multi expert algorithm uses experts in the same way as the WMA and Winnow, however the update of weights is based off of the update used in the dual expert algorithm, in that it is limited by the weight of the selection that is made. The experts used in all three algorithms were of the form

$$a = \begin{cases} 1 & \text{if } n < i \\ 2 & \text{otherwise} \end{cases}$$

Where i is an integer between 1 and M +1. Thus creating 11 control laws, one of which would exactly match the concept prefrence.

Results

Figure 2.3 shows the average total rate of errors. The inclusion of sections noticeably reduces the error rate for DEA. In addition before drift occurs EDEA can be seen to be decreasing quickly, approaching the lowest error rate algorithm, Winnow, along with the Multi Expert algorithm. While after drift occurs Winnow's error rate dramatically increases while WMA, Multi Expert, and Dual expert in each section stay relatively constant. EDEA's error rate also increases, but it still remains less than WMA. This shows that EDEA offers error rate approaching Winnow and the Multi Expert algorithms, and drift resistance that keeps it preforming better than WMA.

Figure 2.4 shows that the number of switches. EDEA and Winnow both have a small rate of switches that decrease towards 0 as time increases before drift. After the drift, the rate of EDEA switches spikes but then resumes descending towards 0. Winnow's switch rate also spikes to about the same height but this is after the same number of trials that were done pre drift, and that extended delay could add to the algorithm feeling unpredictable. In addition the Multi Expert case spikes at the same time as the EDEA, but to a switching rate approximately four times as large.

The continued good performance both before and after drift in the EDEA, and the limited spike in switching at the time of drift supports the fact that it manages the balance between consistency and adaptiveness well.

2.6 Experiment

To test the applicability of the EDEA in an embodied robot we implemented it onto a Turtlebot passing a human pedestrian in the hallway. The parameter was the distance the human was from the right wall. The two selections that the Turtlebot could make was if the human would pass by deflecting to the left or right. It then acted by moving towards the opposite wall.

The Turtlebot was chosen for this experiment since it satisfied the following conditions.

1. The physical dimension of the mobile base occupies a footprint that is similar to the footprint of an averaged human. Hence the robot is able to pass by the bystander in a typical hallway setting.

2. The robot can move at a speed that is comparable to the averaged casual walking speed of a human.

3. The robot is equipped with a kinect and necessary software that can identify an approaching object and its movements to the left or to the right.

5. The robot can be equipped with obstacle avoidance behaviors that can be modified to adjust the avoidance direction.

2.6.1 Setup

The experiments were performed using a single Turtlebot that started centered in the hallway. The Turtlebot continued down the center of the hallway until it detected a human approaching. Then the Turtlebot selected a control law to avoid the human.

While it was moving towards the wall it continued to watch the human to see if its selection was correct. It then used this information to update algorithm 3.

Some limitations were found during the implementation of the program. Most notably the fact that the noise from the depth based human detection made it difficult to identify

human error quickly from a simple jerky motion. It was however usually able to identify errors when the human continued moving towards their intended wall.

Each test was set up with 10 iterations. The human initial location was arbitrarily chosen to get a large coverage of initial positions within tests and different orders of iterations between tests. We placed a marker on the floor so that the human could determine if they started in one of the 6 lanes with a preference towards the left or one of the 4 lanes with a preference towards the right. The tests were repeated for 4 times.

2.6.2 Results

The following table compiles the averaged result over all four tests along with average results from simulation data. It includes the rate of errors, and switches. The lower portion of the table compares the average of the four tests with simulation results for 5%, 10%, and 30% error rates, that were 10 trials long and averaged over 50 tests.

Test	Error Rate	Switch Rate
Experiment average	35.0%	12.5%
Simulation 5% error rate	21.8%	9.8%
Simulation 10% error rate	27.2%	10.7%
Simulation 30% error rate	42.2%	12.1%

The average of the tests' error was within the error bound of the 30% simulation error. While the average number of switching was only slightly higher than the 30 % error bound. This shows even with the difficulties inherent in human perception the EDEA could still have a bounded error and switch rate.

Figure 2.6 shows the averaged weights computed by the robot. This demonstrates the increased difference in weights near the walls which is key for EDEA's consistency, and the decreased difference in weights near the switch point which is key for the EDEA's adaptiveness.

Overall the robot did learn the human's preference which was consistent throughout the tests. The number of errors and switches is consistent with simulation. And the weight ratio displays the same spatial dependence as selected. This means that EDEA transfers well to physical implementation.

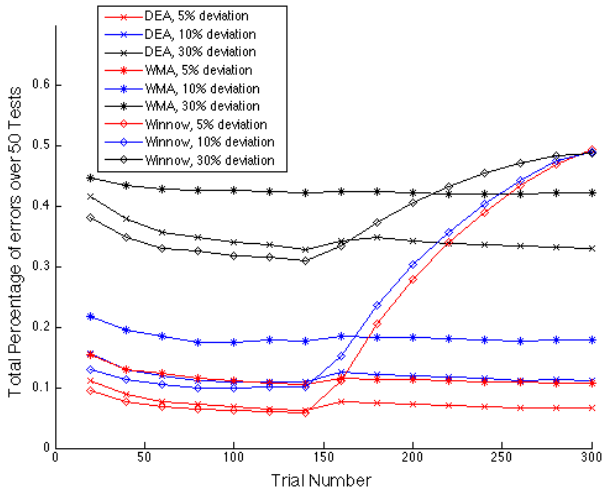

Figure 2.1: Total percentage of errors, averaged over 50 Tests for Duel Expert Algorithm, Weighted Majority Algorithm, and the Winnow Algorithm when the percentage of deviations is 5%, 10%, and 30%, and drift takes place at 150 trials

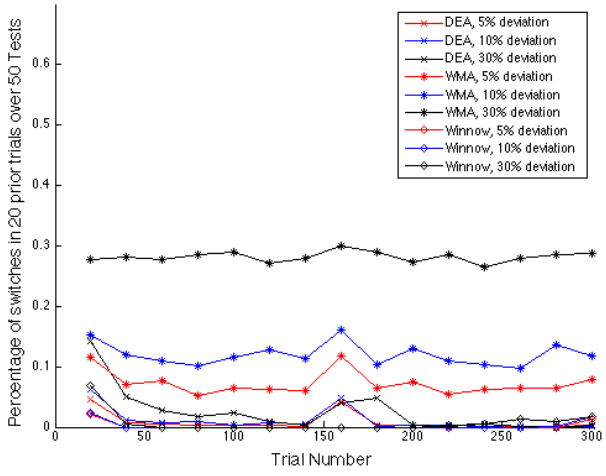

Figure 2.2: Percentage of switches for each group of 20 trials , averaged over 50 Tests for Duel Expert Algorithm, Weighted Majority Algorithm, and the Winnow Algorithm when the percentage of deviations is 5%, 10%, and 30%, and drift takes place at 150 trials

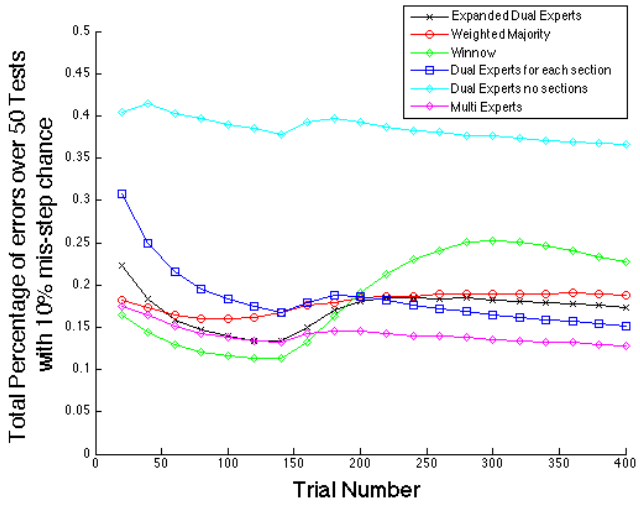

Figure 2.3: Total percentage of incorrect selections, averaged over 50 Tests where the percentage of deviations is 10% and drift occurs at 150 iterations, changing m from 7 to 3.

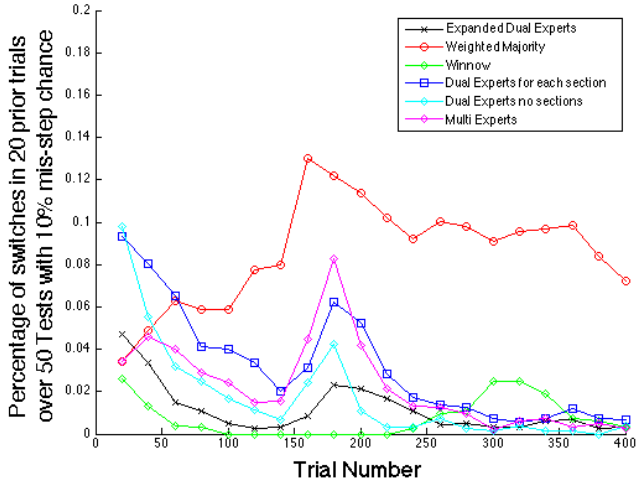

Figure 2.4: Percentage of switches for each group of 20 trials , averaged over 50 Tests where the percentage of deviations is 10% and drift occurs at 150 iterations, changing m from 7 to 3.

Figure 2.5: Turtlebot approaching human starting on the left side of the preference dividing line

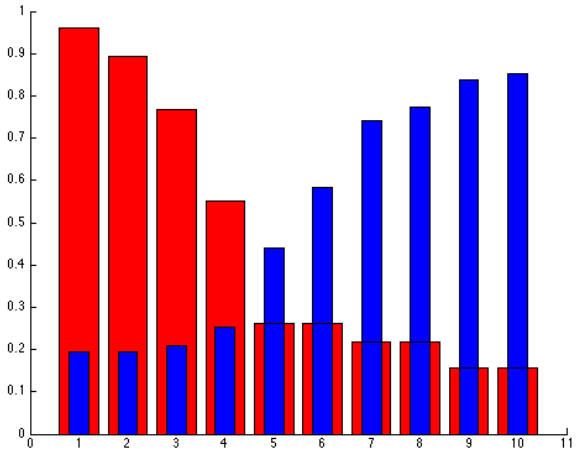

Figure 2.6: Four run average of weights for selection of human turn direction. Hallway position is noted along the x-axis. The red bars represent the weight for the human passing along the left wall while the blue bars represent the weight for the human passing along the right wall .

CHAPTER 3

N-EXPERT ALGORITHM AND MODEL

3.1 Problem Setup

This chapter is designed to show how simple N-expert online learning algorithms can be modeled using Automata. It is an important expansion from the previous chapter because this modeling can be used for a more general analysis that considers N-experts and can be extended to consider cases that do not fall neatly between two already analyzed cases. Online means that each learning algorithm will handle cases sequentially and will not retain previous data sets. And simple N-expert means that each Algorithm will consist of N experts that each predict a unique output.

We make the following assumption regarding the "concept" being learned.

Assumption 3.1.1. *We assume that the concept is a discrete random variable that assumes integer values* $1, 2, ..., N$. *It is then described by a probability distribution* p_i *where* $i = 1, 2, ..., N$. *We also assume that the expert indexed by* i *always select the value of* i *as its prediction.*

We consider four learning algorithms (see Figure 3.1) to learn the probability distribution p_i. The weighted majority algorithm (WMA) and the Winnow algorithm [11, 10] are

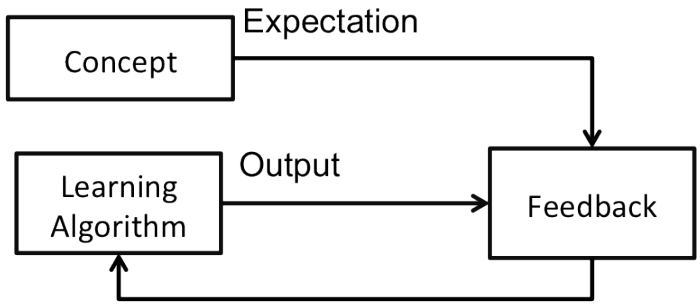

Figure 3.1: Layout of a Learning Algorithm

two classic expert-based learning algorithms. The Multiple Expert Algorithm (MEA) is an extended version of the Dual Expert Algorithm (DEA) considered in the previous chapter. The Human Aware Multiple Expert Algorithm (HAMEA) is a modification of MEA.

The common properties for these algorithms are as follows. N experts are employed, the ith expert will always make the prediction to select the value i for the concept. Hence the probability for the ith expert to make a correct prediction (e.g. being success) is p_i, and the probability for the ith expert to make an error is $\tilde{p}_i = 1 - p_i$. If a wrong prediction is made, then the weight associated with the expert will decrease. The algorithm will select the next prediction by comparing the weights of all experts, and the expert associated with the largest weight will always be selected to make the next prediction. The algorithm generally starts by assigning equal weights to all experts. The three algorithms differ by how they handle weights when a correct prediction is made.

3.2 Four Algorithms

Algorithm 4 The Weighted Majority Algorithm (WMA)

1: Set $W_1 = W_2 = ... = W_N = 0.5$
2: Choose the winner λ as $\min\left(\text{argmax}\left(W_1, W_2, ..., W_N\right)\right)$
3: **if** Error **then**
4: $\quad W_\lambda = \frac{W_\lambda}{2}$
5: **else if** Success **then**
6: $\quad W_\lambda = W_\lambda$
7: **end if**

Algorithm 4 presents the pseudo code for the WMA with N experts. Line 2 indicates that WMA selects the index of the expert with the highest weight. If there are multiple experts having highest weights at an iteration, then WMA selects the one with the smallest index. After the selection is made, WMA updates the weight of the selected expert. If an error occurs, then the weight of the selected expert is divided by 2. If a success occurs, then the weight *remains the same* as the previous iteration.

Remark 3.2.1. *Because all weights start with* 2^{-1} *and can only be decreased by a factor of* 2*, all weights in the WMA will be of the form* 2^{-i} *where* $i \geq 1$ *is an integer.*

Algorithm 5 The Winnow Algorithm

1: Set $W_1 = W_2 = ... = W_N = 0.5$
2: Choose winner λ as $\min(\text{argmax}(W_1, W_2, ..., W_N))$
3: **if** Error **then**
4: $\quad W_\lambda = \frac{W_\lambda}{2}$
5: **else if** Success **then**
6: $\quad W_\lambda = 2W_\lambda$
7: **end if**

Algorithm 5 presents the pseudo code for the Winnow algorithm with N experts. Similar as WMA, the Winnow algorithm selects the expert with the highest weight. If there are multiple experts having the same highest weights, then Winnow selects the one with the smallest index. If an error occurs, then the weight of the selected expert is divided by 2. The difference between WMA and Winnow algorithm is shown by line 6. If a success occurs, i.e. the prediction by an expert is correct, then its weight is multiplied by 2.

Algorithm 6 The Multiple Expert Algorithm (MEA)

1: Set $W_1 = W_2 = ... = W_N = 0.5$
2: Choose output λ from $\min(\text{argmax}(W_1, W_2, ..., W_N))$
3: **if** Error **then**
4: $\quad W_\lambda = \frac{W_\lambda}{2}$
5: **else if** Success **then**
6: $\quad$ **if** $W_\lambda < 0.5$ **then**
7: $\quad\quad W_\lambda = 2W_\lambda$
8: $\quad$ **else**
9: $\quad\quad W_\lambda = W_\lambda$
10: $\quad$ **end if**
11: **end if**

Algorithm 6 presents the pseudo code for the MEA with N experts. The MEA has the same expert selection rule and tie breaker rule as the WMA and winnow algorithms, which is presented by line 2. Also identical to the WMA and the Winnow, if an error occurs, then the weight of the selected expert is divided by 2. The difference is that, as shown by lines 6 and 7, MEA algorithm puts an upper bound on the weights. If a success occurs and the

weight of the selected expert is less than 0.5, then the weight is multiplied by 2. Otherwise, the weight of the selected expert remains the same. Based on these weight updating rules and the initial value of the weights, the weights of MEA are also in the form of 2^i where i is an integer.

Algorithm 7 The Human Aware Multiple Expert Algorithm (HAMEA)

1: Set $W_1 = W_2 = ... = W_N = 0.5$
2: Choose output λ from $\min(\text{argmax}(W_1, W_2, ..., W_N))$
3: **if** Error **then**
4: $W_\lambda = \frac{W_\lambda}{2}$
5: **if** $\lambda \neq \Lambda = \min(\text{argmax}(W_1, W_2, ..., W_N))$ **then**
6: **if** $W_\Lambda \leq 0.25$ **then**
7: $W_\Lambda = 2W_\Lambda$
8: **else**
9: $W_\Lambda = W_\Lambda$
10: **end if**
11: **end if**
12: **else if** Success **then**
13: **if** $W_\lambda < 0.5$ **then**
14: $W_\lambda = 2W_\lambda$
15: **else**
16: $W_\lambda = W_\lambda$
17: **end if**
18: **end if**

Algorithm 7 presents the pseudo code for the HAMEA with N experts. The HAMEA has the same expert selection rule and tie breaker rule as the MEA except in line 6. The difference is that when the algorithm changes outputs an additional weight increase is preformed. The weight corresponding to the new output is multiplied by 2 if that weight is less than 0.5. Based on these weight updating rules and the initial value of the weights, the weights of HAMEA are also in the form of 2^i where i is an integer.

3.3 States

For the four ensemble learning algorithms, the states of the Markov chain models can be described using up to three state variables, λ, R and n. We define λ as the integer index of

the winning expert whose prediction for the concept is also λ. The formula to compute λ is

$$\lambda = \min\left(\text{argmax}\left(W_1, W_2, ..., W_N\right)\right). \tag{3.1}$$

The state variable R is defined as the ratio between the maximal weight and the minimal weight among all experts. The formula to compute R is

$$R = \begin{cases} \log_2\left(\frac{2\max\left(W_1,...,W_N\right)}{\min\left(W_1,...,W_N\right)}\right), & \text{if } \lambda = 1 \\ \log_2\left(\frac{\max\left(W_1,...,W_N\right)}{\min\left(W_1,...,W_N\right)}\right), & \text{otherwise} \end{cases} \tag{3.2}$$

When the prediction $\lambda = 1$, we multiply the maximal weight with 2 to normalize the ratio R. Because all the weights are in the form of 2^{-i}, R is an integer. Since the ratio $\frac{\max\left(W_1,...,W_N\right)}{\min\left(W_1,...,W_N\right)}$ is greater than or equal to 1, then $R \geq 0$.

To model the MEA and HAMEA, we also need to introduce another state variable n as

$$n = \begin{cases} \log_2\left(\frac{1}{\min\left(W_1,...,W_N\right)}\right), & \text{if } \lambda = 1 \\ \log_2\left(\frac{0.5}{\min\left(W_1,...,W_N\right)}\right), & \text{otherwise} \end{cases} \tag{3.3}$$

Here n represents the maximal achievable value of R. Because the weights in MEA and HAMEA have an upper bound 0.5, replacing the term $\max\left(W_1, ..., W_N\right)$ in equation (3.2) with 0.5 results in equation (3.3). Based on this definition, $R \leq n$ always holds.

3.3.1 N-expert Weighted Majority Algorithm

In the following proposition, we will show that for WMA, the value of R can only be 1.

Proposition 3.3.1. *The variable R equals 1 for all $\lambda = 1, ..., N$ for WMA.*

Proof. We will prove this proposition by contradiction. First, we assume that R can be 0. The only case where $R = 0$ is when $\lambda \neq 1$ and $\frac{\max\left(W_1,...,W_N\right)}{\min\left(W_1,...,W_N\right)} = 1$, which means that $W_1 = W_2 = \cdots = W_N$. However, since all the weights are the same, based on line 2 of

Algorithm 4, the index of the winning expert λ should be 1, which contradicts to the fact that $\lambda \neq 1$. Therefore, we have shown that $R > 0$.

Secondly, we assume that R can be 2. Due to equation (3.2), we need to discuss two cases. **Case 1**: $\lambda = 1$. In this case, the maximal weight is W_1. If $R = 2$, then we have $W_1 = 2\min(W_2, ..., W_N)$. Denote the smallest index of the minimal weights among $(W_2, ..., W_N)$ to be j. We have $W_1 = 2W_j$ and $j > 1$. Since the initial weights are all identical and the weight of an expert can only be decreased if this expert is selected by WMA at one iteration, $W_1 = 2W_j$ can only occur if the expert j is selected as the winning expert at one previous iteration and then be divided by 2. This means that the weights of experts 1 and j are the same but the algorithm selects expert j with $j > 1$, which contradicts to the tie breaker of WMA, i.e., line 2 in Algorithm 4. **Case 2**: $\lambda \neq 1$. In this case, we denote the smallest index of the maximal weights among $(W_2, ..., W_N)$ to be j_1 and the smallest index of the minimal weights among $(W_1, ..., W_N)$ to be j_2. Since we assume $R = 2$, we have $W_{j_1} = 4W_{j_2}$. Since the initial weights are all identical and the weight of an expert can only be decreased if this expert is selected by WMA at one iteration, $W_{j_1} = 4W_{j_2}$ can only occur if the expert j_2 is the winning expert at one previous iteration and then be divided by 2. This means that WMA selects expert j_2 with $W_{j_1} = 2W_{j_2}$ at one previous iteration, which violates line 2 in Algorithm 4. Therefore, by contradiction, we have shown that $R \neq 2$. Using similar argument, we can also show that R cannot be greater than 2.

In conclusion, R is an integer that must satisfy $R > 0$ and $R < 2$, which means that R can only be equal to 1. □

Based on Proposition 3.3.1, we can derive two corollaries which present the relations among the weights of all the experts. The corollaries will help to determine the structure of the Markov chain of WMA.

Corollary 3.3.2. *For WMA, if the index of the winning expert λ is 1, then we have $W_1 = W_2 = \cdots = W_N$.*

Proof. Based on equation (3.2), we have

$$R = \log_2 \left(\frac{2 \max (W_1, ..., W_N)}{\min (W_1, ..., W_N)} \right) = 1,$$

which means that $\frac{\max (W_1, ..., W_N)}{\min (W_1, ..., W_N)} = 1$, i.e., the maximal weights equal the minimal weights. This shows that all the weights are the same. □

Corollary 3.3.3. *For WMA, if the index of the winning expert λ is greater than 1, then we have $W_1 = \cdots = W_{\lambda-1} = \frac{1}{2} W_\lambda = \cdots = \frac{1}{2} W_N$.*

Proof. Since the index of the winning expert is λ, it means that for any $j \in \{1, ..., \lambda - 1\}$, $W_j < W_\lambda$ must be satisfied. Otherwise, if there exists an index $j < \lambda$ and $W_j \geq W_\lambda$, then the winning expert should be j. Based on equation (3.2), we have

$$R = \log_2 \left(\frac{W_\lambda}{\min (W_1, ..., W_N)} \right) = 1,$$

which means that $\frac{1}{2} W_\lambda = \min (W_1, ..., W_N)$. Based on Remark 3.2.1, if a weight W_j is strictly smaller than W_λ, then $W_j \leq \frac{1}{2} W_\lambda$. Therefore, we have $\frac{1}{2} W_\lambda = \min (W_1, ..., W_N) \leq W_j \leq \frac{1}{2} W_\lambda$ for $j \in \{1, ..., \lambda - 1\}$, which can only hold if $W_1 = \cdots = W_{\lambda-1} = \frac{1}{2} W_\lambda$. If $\lambda = N$, then the proof is complete.

If $1 < \lambda < N$, then we need to show that $W_\lambda = \cdots = W_N$. We also show this by contradiction. Because λ is the index of the maximal weight, we have $W_\lambda \geq W_j$ for $j \in \{\lambda + 1, ..., N\}$. If there exists an index $j \in \{\lambda + 1, ..., N\}$ such that $W_j < W_\lambda$, then $W_j = \frac{1}{2} W_\lambda$ since $R = 1$. Because all the initial weights are 0.5 and the weight of an expert can only be decreased if this expert is the winning expert at one iteration, $W_j = \frac{1}{2} W_\lambda$ can only occur if the expert j is selected previously and then be divided by 2. This means that the weights of experts λ and j are the same but the algorithm selects expert j as he winning expert with $j > \lambda$, which contradicts to the line 2 in Algorithm 4. Therefore, for any $j \in \{\lambda + 1, ..., N\}$, $W_j = W_\lambda$. □

Utilize the two variables defined by equations (3.1) and (3.2), we can construct the Markov chain of WMA. Since R is always 1, the total number of the states of the WMA Markov chain is N. The states are $[\lambda, 1]$ where $\lambda = 1, ..., N$. We will first show how states transit when a success occurs.

Lemma 3.3.4. *For all states* $(\lambda, 1)$ *where* $\lambda \in \{1, ..., N\}$*, if a success occurs, then the state remains in* $(\lambda, 1)$.

Proof. Line 5 in Algorithm 4 shows that the weight of the winning expert does not change if a success occurs. Therefore, if a success occurs, the index of the winning expert remains the same at the next selection. □

Next we will show how the states of the Markov chain transit when an error occurs.

Lemma 3.3.5. *For all states* $(\lambda, 1)$ *where* $\lambda \in \{1, ..., N-1\}$*, if an error occurs, then the state transits to state* $(\lambda + 1, 1)$.

Proof. Because of equation (3.2), we need to discuss two cases. **Case 1**: $\lambda = 1$. Based on Corollary 3.3.2, we have $W_1 = W_2 = \cdots = W_N$. From line 3, if an error occurs, then W_1 is divided by 2. At the next selection, the decreased weight W_1 satisfies $W_1 = \frac{1}{2}W_2 = \cdots = \frac{1}{2}W_N$. Then based on line 2, the index of the winning expert will be 2, which makes the state transit from $(1, 1)$ to $(2, 1)$.

Case 2: $\lambda \neq 1$. Based on Corollary 3.3.3, we have $W_1 = \cdots = W_{\lambda-1} = \frac{1}{2}W_\lambda = \cdots = \frac{1}{2}W_N$. If an error occurs, then W_λ is divided by 2. At the next selection, the decreased weight W_λ satisfies $W_1 = \cdots = W_\lambda = \frac{1}{2}W_{\lambda+1} = \cdots = \frac{1}{2}W_N$. Then based on line 2, the index of the winning expert will be $\lambda + 1$, which makes the state transit from $(\lambda, 1)$ to $(\lambda + 1, 1)$. □

Lemma 3.3.6. *For the state* $(N, 1)$*, if an error occurs, then the state transits to* $(1, 1)$.

Proof. Based on Corollary 3.3.3, we have $W_1 = \cdots = W_{N-1} = \frac{1}{2}W_N$. If an error occurs, then W_N is divided by 2. At the next selection, the decreased weight W_N satisfies $W_1 =$

$\cdots = W_N$. Then based on line 2, the index of the winning expert will be 1, which makes the state transit from $(N, 1)$ to $(1, 1)$. □

To summarize, Lemma 3.1.1 presents the transition probabilities for each success and error. Lemma 3.3.4 presents how the states transit when a success occurs. Lemma 3.3.5 and Lemma 3.3.6 present how the states transit when an error occurs. Based on these Lemmas, the Markov chain for WMA can be constructed as shown in Figure 3.2.

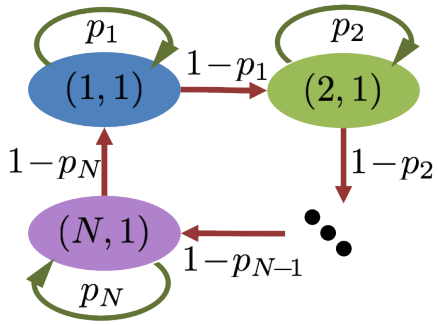

Figure 3.2: Markov chain for simple N-Expert WMA.

The full transition matrix $\mathbb{P}_{WMA}$ of WMA is

$$\mathbb{P}_{WMA} = \begin{bmatrix} p_1 & \widetilde{p}_1 & 0 & \dots & 0 \\ 0 & p_2 & \widetilde{p}_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \widetilde{p}_N & 0 & 0 & \dots & p_N \end{bmatrix}. \tag{3.4}$$

3.3.2 N-expert Winnow Algorithm

We use the same variables λ and R in equations (3.1) and (3.2) to describe the states of the Markov chain of the Winnow algorithm. And same as WMA, the probability for a success to occur is p_λ and the probability for an error to occur is $\widetilde{p}_\lambda$. All weights in Winnow algorithm are of the form 2^{-i} where i is an integer. Different from WMA, i can be negative, which means that the weight of an expert can have an unbounded increase. Therefore, the variable $R \geq 1$ for Winnow is not always equal to 1 but can be infinitely large.

The following proposition reveal the relations among the weights in Winnow algorithm.

Proposition 3.3.7. *For Winnow, if the index of the winning expert λ is 1, then we have $W_2 = \cdots = W_N$. If the index of the winning expert λ is greater than 1, then we have $W_1 = \cdots = W_{\lambda-1} = \frac{1}{2}W_{\lambda+1} = \cdots = \frac{1}{2}W_N$.*

Proof. Since the initial weights are all equal, the expert 1 will be the winning expert at the first selection step t_0 and the weight W_1 will change after the first selection. If a success occurs, then W_1 will be multiplied with 2. If an error occurs, then W_1 will be divided by 2. Let t_1 be the first selection step when $W_1 = \frac{1}{2}W_2 = \cdots = \frac{1}{2}W_N$. Within steps $[t_0, t_1)$, expert 1 is always the winning expert and the weight W_1 is the only weight whose value is changed since $W_1 \geq W_2 = \cdots = W_N$. All the other weights are equal to the initial weight. The weights satisfy $W_2 = \cdots = W_N$ with the interval $[t_0, t_1)$.

At selection step t_1, because $W_2 = 2W_1 > W_1$ and $W_2 = \cdots = W_N$, the expert 2 will be the winning expert. Let t_2 be the first selection step when $W_2 = \frac{1}{2}W_3$. Within steps $[t_1, t_2)$, expert 2 is the winning expert and the weight W_2 is the only weight whose value is changed. The weights satisfy $W_3 = \cdots = W_N$ and $W_1 = \frac{1}{2}W_3$ with the interval $[t_1, t_2)$, i.e. $W_1 = \frac{1}{2}W_3 = \cdots = \frac{1}{2}W_N$.

Similar, we can define $t_\lambda, \lambda = 1, ..., N-1$ be the first selection step when $W_\lambda = \frac{1}{2}W_{\lambda+1}$. Within steps $[t_{\lambda-1}, t_\lambda)$, expert λ is the winning expert and the weight W_λ is the weight whose value is changed. The weights satisfy $W_1 = \cdots = W_{\lambda-1} = \frac{1}{2}W_{\lambda+1} = \cdots = \frac{1}{2}W_N$ with the interval $[t_{\lambda-1}, t_\lambda)$.

After selection step t_N, all the weights are all equal. Then the weights updating will repeat the above process. □

Similar to WMA, we will first show how the states of Markov chain of Winnow transit when a success occurs.

Lemma 3.3.8. *For all states $[\lambda, R]$, if a success occurs, then the state transits to $[\lambda, R+1]$.*

Proof. Since the current state is $[\lambda, R]$, the weight W_λ is greater than or equal to all the other weight W_j where $j \neq \lambda$ with $W_\lambda = 2^R \min_{j\in\{1,...,N\}\setminus\lambda} W_j$. If a success occurs, then by line 5 of algorithm 5, we know updated weight of expert λ, denoted as W'_λ, equals $2W_\lambda$. Therefore, W'_λ is greater than all the other weight W_j where $j \neq \lambda$. Then at the next iteration, the index of the winning expert should still be λ. And $W'_\lambda = 2^{R+1} \min_{j\in\{1,...,N\}\setminus\lambda} W_j$, which makes the state transit to $[\lambda, R+1]$. □

Based on Proposition 3.3.7, we will show how the states of the Markov chain of Winnow transit when an error occurs.

Lemma 3.3.9. *For all states (λ, R) with $R \geq 2$, if an error occurs, then the state transits to state $(\lambda, R-1)$.*

Proof. Based on Proposition 3.3.7, we have $W_1 = 2^{R-1}W_j$ where $j \in \{2, ..., N\}$. If an error occurs, then by line 3 of algorithm 5, W_1 is decreased by a factor of 2. Then the newly updated weight $W_1 = 2^{R-2}W_j$ for $j \in \{2, ..., N\}$. Since $R \geq 2$, the coefficient $2^{R-2} \geq 1$, which means that $W_1 \geq W_j$ for $j \in \{2, ..., N\}$. At the next selection, expert 1 will be selected with R reduced by 1, which makes the new state $(1, R-1)$.

If $\lambda = N$, then we have $W_N = 2^R \min(W_1, ..., W_{N-1})$. Based on Proposition 3.3.7, we have $W_N = 2^R W_j$ where $j \in \{1, ..., N-1\}$. If an error occurs, W_N is decreased by a factor of 2. Then the newly updated weight $W_N = 2^{R-1}W_j$ for all $j < N$. Since $R \geq 2$, $W_N \geq 2W_j$ for all $j < N$. At the next selection, expert N will be selected with R reduced by 1, which makes the new state $(N, R-1)$.

If $\lambda \neq 1$ and $\lambda < N$, we have $W_1 = 2^R \min(W_1, ..., W_N)$. Based on Proposition 3.3.7, we have $W_1 = 2^R W_j$ for $j < \lambda$ and $W_1 = 2^{R-1}W_j$ for $j > \lambda$. If an error occurs, W_λ is decreased by a factor of 2. Then the newly updated weight $W_\lambda = 2^{R-2}W_j$ for $j > \lambda$. Since $R \geq 2$, $W_\lambda \geq W_j$ for all $j \neq \lambda$. At the next selection, expert λ will be selected with R reduced by λ, which makes the new state $(\lambda, R-1)$. □

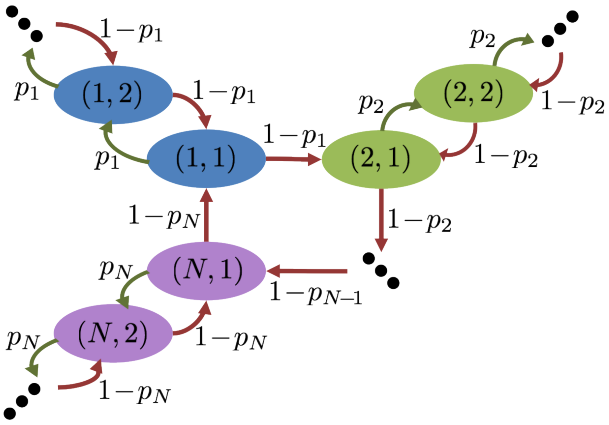

Figure 3.3: Markov chain for simple N-Expert Winnow Algorithm.

Lemma 3.3.10. *For all states* (λ, R) *with* $\lambda \in \{1...N-1\}$ *and* $R = 1$*, if an error occurs, then the state transits to* $(\lambda + 1, 1)$.

Proof. This is the same case as Lemma 3.3.5. □

Lemma 3.3.11. *For the state* $(N, 1)$*, if an error occurs, then the state transits to* $(1, 1)$.

Proof. This is the same case as Lemma 3.3.6. □

In summary, Lemma 3.1.1 gives the transition probabilities for each success and error. Lemma 3.3.8 gives the update when a success occurs. Lemma3.3.9, 3.3.10, and 3.3.11 gives the update when an error occurs. The Markov chain of Winnow is shown by Figure 3.3.

3.3.3 Multi-Expert Algorithm

We use variables λ, R and n to model the states of Markov chain for MEA. We first present how states transits if a success occurs.

Lemma 3.3.12. *For any state* (λ, R, n) *with* $R < n$*, if a success occurs, then the state transits to state* $(\lambda, R + 1, n)$.

Proof. If $R < n$, then we have $W_\lambda < 0.5$. And since λ is the winning expert, the weight W_λ is greater than or equal to all the other weights. If a success occurs, then by line 7 of

algorithm 6, then W_λ will be doubled and the updated weight W_λ will be greater than all the other weights. At the next selection, λ will be the winning expert. And since the only changed weight is W_λ, the value of $\min(W_1, ..., W_N)$ is unchanged. Therefore, n does not change and R is increased by 1, which makes the state transit to state $(\lambda, R+1, n)$. □

Lemma 3.3.13. *For any (λ, R, n) with $R = n$, if a success occurs, then the state remains at (λ, R, n).*

Proof. If $R = n$, then we have $W_\lambda = 0.5$. If a success occurs, then by line 9 of algorithm 5, then W_λ will remain the same. With all the weights unchanged at the next selection, the state also remains the same. □

Lemma 3.3.14. *For all states (λ, R, n) with $R \geq 2$, if an error occurs, then the state transits to $(\lambda, R-1, n)$*

Proof. The variables λ and R follow the same proof as Lemma 3.3.9. For the variable n, since the only changed weight is W_λ and the updated W_λ is still greater than or equal to the other weights, the value of $\min(W_1, ..., W_N)$ is unchanged. Therefore, the value of n also does not change. □

Lemma 3.3.15. *For all states (λ, R, n) with $\lambda \in \{1...N-1\}$ and $R = 1$, if an error occurs, then the state transits to $(\lambda + 1, 1, n)$.*

Proof. The variables λ and R follow the same proof as Lemma 3.3.10. For the variable n, if $\lambda = 1$, then the updated W_1' equals $\frac{1}{2}\min(W_1, ..., W_N)$. Then after the next selection, with $\lambda = 2$ and $\min(W_1', ..., W_N) = \frac{1}{2}\min(W_1, ..., W_N)$, the value of n does not change because the numerator and denominator of equation (3.3) are both divided to its half. If $\lambda > 1$, then the updated W_λ equals $\min(W_1, ..., W_N)$. Because of $\lambda > 1$ and $lambda+1 > 1$, the numerator in equation (3.3) remains 0.5. Therefore, the value of n remains the same. □

Lemma 3.3.16. *For states $(N, 1, n)$, if an error occurs, then the state transits to $(1, 1, n+1)$.*

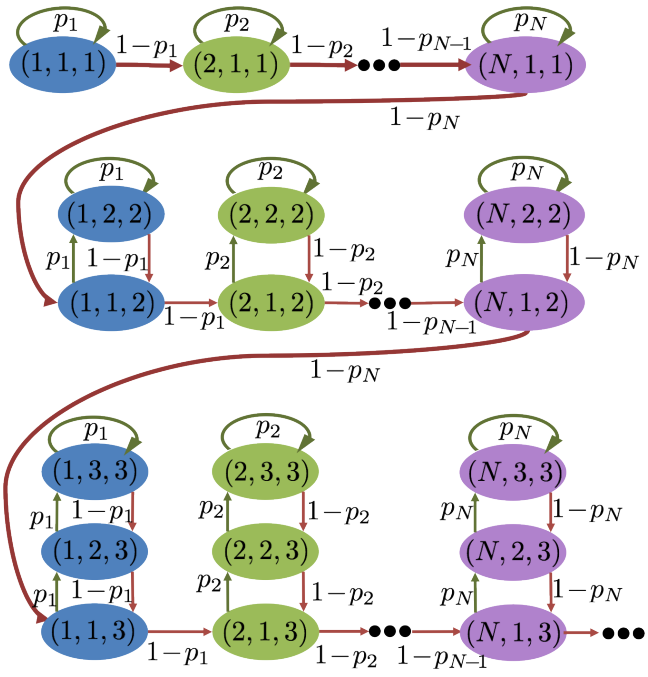

Figure 3.4: Markov chain for simple N-Expert MEA.

Proof. If $\lambda = N$, then $W_N > W_j$ for $j \in \{1, ..., N-1\}$. In addition, since $R = \log_2\left(\frac{\max(W_1 ... W_N)}{\min(W_1 ... W_N)}\right) = 1$ which implies $W_N = 2\min(W_1, ..., W_N)$. Combining these two results, we have $W_N = 2W_1 = ... = 2W_{N-1}$. Then if an error occurs, W_N is divided by 2, resulting in the updated weight W'_N. The updated weight W'_N satisfies $W'_N = W_1 = \cdots = W_{N-1}$. And imputing this into equations (3.1) and (3.2), we have $\lambda = 1$ and $R = 1$. For variable n, the value of $\min(W_1 ... W'_N)$ equals $\min(W_1 ... W_N)$. However, since the winning expert changes to be $\lambda = 1$, the numerator in equation (3.3) remains 1 instead of 0.5. Therefore, the value n becomes $n + 1$, resulting in the transition from state $(N, 1, n)$ to state $(1, 1, n + 1)$. □

The Markov chain for MEA can be constructed as shown by Figure 3.4. As we can see, the strucure of MEA is more complicated than WMA and Winnow because MEA has relatively most complicated updating rules for weights.

3.3.4 Human Aware Multi-Expert Algorithm

We use variables λ, R and n to model the states of Markov chain for HAMEA. We first present how states transits if a success occurs.

Lemma 3.3.17. *For any state (λ, R, n) with $R < n$, if a success occurs, then the state transits to state $(\lambda, R+1, n)$.*

Proof. This is the same case as Lemma 3.3.12. □

Lemma 3.3.18. *For any (λ, R, n) with $R = n$, if a success occurs, then the state remains at (λ, R, n).*

Proof. This is the same case as Lemma 3.3.13. □

Lemma 3.3.19. *For all states (λ, R, n) with $R \geq 2$, if an error occurs, then the state transits to $(\lambda, R-1, n)$*

Proof. This is the same case as Lemma 3.3.14. □

Lemma 3.3.20. *For all states $(\lambda, R, 1)$ with $\lambda \in \{1...N-1\}$ and $R = 1$, if an error occurs, then the state transits to $(\lambda + 1, 1, 1)$.*

Proof. From Lemma 3.3.15 we know that before HAMEA enacts the update on line 6 the state would be $(\lambda + 1, 1, 1)$. Then since $n = 1$ and $\lambda + 1 > 1$ equation 3.3 shows $\min W_1...W_N = 0.25$. Since $R = 1$ equation 3.2 shows $\max W_1...W_N = 0.5$. Thus the algorithm follows line 8 and the final state remains $(\lambda + 1, 1, 1)$. □

Lemma 3.3.21. *For all states (λ, R, n) with $\lambda \in \{1...N-1\}$, $R = 1$ and $n \geq 2$ if an error occurs, then the state transits to $(\lambda + 1, 2, n)$.*

Proof. From Lemma 3.3.15 we know that before HAMEA enacts the update on line 6 the state would be $(\lambda + 1, 1, n)$. Then since $n > 1$ and $\lambda + 1 > 1$ equation 3.3 shows $\min W_1...W_N < 0.25$. Since $R = 1$ equation 3.2 shows $\max W_1...W_N \leq 0.25$. Thus the

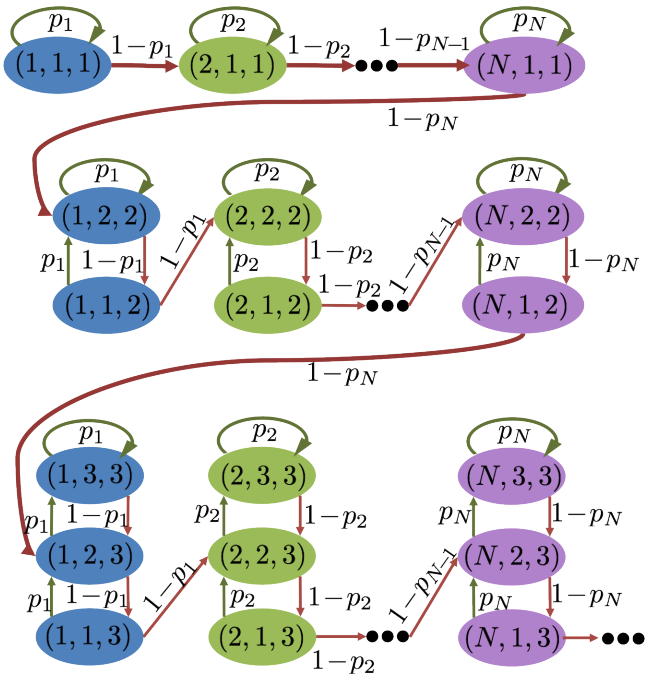

Figure 3.5: Markov chain for simple N-Expert HAMEA.

algorithm follows line 5 and $\max W_1 ... W_N$ is multiplied by 2. This means that the $R = 2$, and the state becomes $(\lambda + 1, 2, n)$. □

Lemma 3.3.22. *For states* $(N, 1, n)$*, if an error occurs, then the state transits to* $(1, 2, n + 1)$.

Proof. From Lemma 3.3.16 we know that before HAMEA enacts the update on line 6 the state would be $(1, 1, n + 1)$. Then since $n > 1$ equation 3.3 shows $\min W_1 ... W_N \leq 0.25$. Since $R = 1$ equation 3.2 shows $\max W_1 ... W_N \leq 0.25$. Thus the algorithm follows line 5 and $\max W_1 ... W_N$ is multiplied by 2. This means that the $R = 2$, and the state becomes $(1, 2, n + 1)$. □

The Markov chain for HAMEA can be constructed as shown by Figure 3.5. As we can see, the structure of HAMEA is similar to that of MEA, with the only difference being in the transitions when λ changes.

CHAPTER 4

CONSISTENCY AND ADAPTIVENESS

4.1 Problem Setup

The goal of this chapter is to provide quantitative evaluation of the adaptiveness and consistency of the four ensemble learning algorithms. We now introduce necessary mechanisms to formulate the problems.

Let α be the index where p_α is the maximum value among all indices $i = 1, 2, ..., N$. The value α is called a preference of the concept. We consider this preference as a strong preference, which is described by the following assumption:

Assumption 4.1.1. *We assume that* $p_\alpha > p_i, \forall i \neq \alpha$ *and* $p_\alpha > 0.5$.

The assumption indicates that α is unique. For convenience of notations, we rename the indices of the $(N-1)$ experts that are not α as $\beta_1, ..., \beta_{N-1}$ where $\beta_1 = \alpha + 1$ and $\beta_{N-1} = \alpha + N - 1$. The probabilities of all the experts satisfy that $p_\alpha + \sum_{i=1}^{N-1} p_{\beta_i} = 1$. We can rewrite this equality as $1 - p_\alpha = \sum_{i=1}^{N-1} p_{\beta_i}$. Since $p_{\beta_i} \geq 0$ for any $i = 1, ..., N-1$, we have $1 - p_\alpha = \sum_{i=1}^{N-1} p_{\beta_i} \geq p_{\beta_i} \geq 0$.

Remark 4.1.2. *The assumption requires that a concept has a strong preference for value* α*. This assumption is usually required for ensemble learning to perform well. Under this assumption, the learning algorithms will eventually select expert* α *as the winning expert more often than other experts.*

The problem for evaluating adaptiveness for an ensemble learning algorithm can then be formulated as follows:

Problem 4.1.3. *(evaluating adaptiveness) Suppose the winning expert is currently not* α*, calculate the averaged number of iterations before an algorithm chooses* α *as the winning expert.*

Remark 4.1.4. *Under this problem formulation, adaptiveness is measured by how many iterations for the algorithm to learn the strong preference for the concept. If the number of iterations is small, then the algorithm can quickly learn the preference. This is especially preferred when the value of α suddenly changes. The algorithm can adapt to this change quickly.*

On the other hand, the problems for evaluating consistency for an ensemble learning algorithm can be formulated as follows:

Problem 4.1.5. *(evaluating consistency) Suppose the winning expert is currently α, calculate the averaged number of iterations before an algorithm chooses another expert, different from α, as the winning expert.*

Remark 4.1.6. *Under this problem formulation, consistency is measured by how many iterations can the algorithm tolerate before it chooses a winning expert that is not the strong preference of the concept. If the number of iterations is large, then the algorithm can tolerate large but temporary deviations from the preference.*

The two problems can be solved by numerical simulations. However, numerical simulations can only provide answer on a case by case basis, and often fail to offer insights of the solutions. In this chapter analytic solutions to the two problems based on Markov chain models for the four ensemble learning algorithms are proposed.

4.2 Adaptiveness and Consistency Analysis

We are now ready to provide the analytic solutions for the problems of evaluating adaptiveness and consistency for the four learning algorithms. The strong preference α represents the states in the Markov chain modes where $\lambda = \alpha$. Then the number of iterations to transit to or from these states can be calculated as the mean hitting steps between certain subsets of states on the Markov chains.

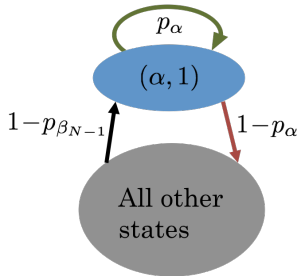

Figure 4.1: The partial Markov chain for N-Expert WMA. The blue circle represents the set A and the grey circle represents the set B.

4.2.1 Mean Hitting Steps

Let the function $h_A(k)$ be defined as the mean hitting steps [41] for the state of a Markov chain starting from state k and eventually move to the subset A that contains some states. Obviously $h_A(k) = 0$, if the state k belongs to A. Let $\mathbb{P}$ be the transition matrix associated with the Markov chain. The number of rows and columns in $\mathbb{P}$ corresponding to the number of states in the Markov chain. After all the rows and columns corresponding to the states in A have been removed from the transition matrix $\mathbb{P}$ of the Markov chain, we obtain a sub-matrix $P_{/A}$, called the partial transition matrix. Let $\mathbf{h}_A$ be the column vector formed by all the values of $h_A(k)$ for all the state k that are not in A. Then the mean hitting steps can be calculated by

$$\mathbf{h}_A = (I - P_{/A})^{-1}\mathbf{1} \tag{4.1}$$

where I is the identity matrix, and $\mathbf{1}$ is a column vector with each element equals to 1. The dimension of $\mathbf{h}_A$ and $\mathbf{1}$ equals to the number of states that are not in the subset A.

For our purpose, we define the subset A as the set which contains all the states where $\lambda = \alpha$. We also define the subset B as the set which contains all the states where $\lambda \neq \alpha$. Then the set $A \bigcup B$ contains all the states in a Markov chain that models one of the three algorithms. We will first construct the partial transition matrices $P_{/A}$ and $P_{/B}$, and then compute the mean hitting times $\mathbf{h}_A$ and $\mathbf{h}_B$ using equation (4.1), which will be used as the metric for adpativeness and consistency analysis.

For WMA, A contains state $(\alpha, 1)$. After A is removed, the partial transitions matrix is

$$P_{/A} = \begin{bmatrix} p_{\beta_1} & \widetilde{p}_{\beta_1} & 0 & \dots & 0 \\ 0 & p_{\beta_2} & \widetilde{p}_{\beta_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & p_{\beta_{N-1}} \end{bmatrix}. \tag{4.2}$$

Based on equations (4.1) and (4.2), we can compute the mean hitting steps from states in the set B to the set A as

$$\begin{aligned} \mathbf{h}_A &= \begin{bmatrix} 1-p_{\beta_1} & -\widetilde{p}_{\beta_1} & 0 & \dots & 0 \\ 0 & 1-p_{\beta_2} & -\widetilde{p}_{\beta_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1-p_{\beta_{N-1}} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{p}_{\beta_1} & -\widetilde{p}_{\beta_1} & 0 & \dots & 0 \\ 0 & \widetilde{p}_{\beta_2} & -\widetilde{p}_{\beta_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \widetilde{p}_{\beta_{N-1}} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\widetilde{p}_{\beta_1}} & \frac{1}{\widetilde{p}_{\beta_2}} & \frac{1}{\widetilde{p}_{\beta_3}} & \dots & \frac{1}{\widetilde{p}_{\beta_{N-1}}} \\ 0 & \frac{1}{\widetilde{p}_{\beta_2}} & \frac{1}{\widetilde{p}_{\beta_3}} & \dots & \frac{1}{\widetilde{p}_{\beta_{N-1}}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\widetilde{p}_{\beta_{N-1}}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \\ \sum_{i=2}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \\ \vdots \\ \sum_{i=N-1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \end{bmatrix} \end{aligned} \tag{4.3}$$

In the mean hitting steps, we will take the first element in $\mathbf{h}_A$ as the metric to quantify adaptiveness, which is

$$\mathbf{h}_A(1) = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \tag{4.4}$$

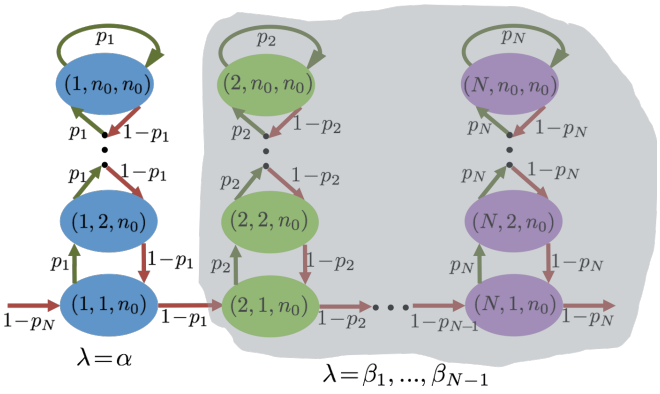

Figure 4.2: Partial Markov chains of MEA with $n = n_0$. The blue circles represent the states in set A_{n_0} with preferred output and the circles in the grey area represent the states in set B_{n_0}.

For WMA, after B is removed the only state left is $(\alpha, 1)$. Then $P_{/B}$ is an 1 by 1 matrix as

$$P_{/B} = \left[p_\alpha \right]. \tag{4.5}$$

Based on equations (4.1) and (4.5), we can compute the mean hitting steps from the state $(\alpha, 1)$ to the set B as

$$\mathbf{h}_B = \frac{1}{1 - p_\alpha} = \frac{1}{\widetilde{p}_\alpha}. \tag{4.6}$$

Since Winnow is a special case of MEA when n goes to infinity, we will first presents formula to the meaning hitting steps for MEA and then present the results for Winnow by letting n go to infinity.

For each n, we define A_n to be the set that contains all the states (α, R, n) and B_n to be the set that contains all the states (λ, R, n) where $\lambda \neq \alpha$. The sets A_n and B_n with $n = n_0$ are illustrated by Figure 4.2. For an n, we will first introduce the partial transition matrix

P_i^n for the partial Markov chain consisting of states $[i, R, n]$ for fixed i and n.

$$P_i^n = \begin{bmatrix} 0 & p_i & 0 & \dots & 0 & 0 & 0 \\ 1-p_i & 0 & p_i & \dots & 0 & 0 & 0 \\ 0 & 1-p_i & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & p_i & 0 \\ 0 & 0 & 0 & \dots & 1-p_i & 0 & p_i \\ 0 & 0 & 0 & \dots & 0 & 1-p_i & p_i \end{bmatrix} \tag{4.7}$$

where P_i^n is an $n \times n$ matrix. We also define a matrix C_i^n to be

$$C_i^n = \begin{bmatrix} 1-p_i & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{4.8}$$

which is also an $n \times n$ matrix. Then the partial transition matrix $\mathbb{P}^n$ for the partial Markov chain consisting of states (i, R, n) for $i = 1, ..., N$ under a fixed n is

$$\mathbb{P}^n = \begin{bmatrix} P_1^n & C_1^n & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & P_2^n & C_2^n & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & P_3^n & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & P_N^n \end{bmatrix} \tag{4.9}$$

where $\mathbf{0}$ is an $n \times n$ matrix with all elements equal to 0.

After A_n is removed, the partial transitions matrix is

$$P^n_{/A_n} = \begin{bmatrix} P^n_{\beta_1} & C^n_{\beta_1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & P^n_{\beta_2} & C^n_{\beta_2} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & P^n_{\beta_3} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & P^n_{\beta_{N-1}} \end{bmatrix} \tag{4.10}$$

Based on equation (4.1), to compute the mean hitting time from states in the set B_n to the set A_n, we need to compute $\left(I - P^n_{/A_n}\right)^{-1}$. In order to show this we first need to show following lemmas.

Lemma 4.2.1. *Every element in the first column of* $\left(I - P^n_{\beta_i}\right)^{-1}$ *is* $\frac{1}{1-p_{\beta_i}}$.

Proof. If $n = 1$, then $I - P^n_\beta = 1 - p_\beta$. Thus the only element in $\left(I - P^n_{\beta_i}\right)^{-1}$ is $\frac{1}{1-p_{\beta_i}}$.

If $n = 2$, then we have

$$\left(I - P^n_{\beta_i}\right)^{-1} = \begin{bmatrix} 1 & -p_{\beta_i} \\ p_{\beta_i} - 1 & 1 - p_{\beta_i} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{1-p_{\beta_i}} & \frac{p_{\beta_i}}{(1-p_{\beta_i})^2} \\ \frac{1}{1-p_{\beta_i}} & \frac{1}{(1-p_{\beta_i})^2} \end{bmatrix}.$$

For $n > 2$, the inverse of the matrix $\left(I - P^n_{\beta_i}\right)$ is not simple to compute. Let us define

$$X = \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \dots & x_{n,n} \end{bmatrix} = \left(I - P^n_{\beta_i}\right)^{-1} \tag{4.11}$$

Based on the definition of an inverse, we have $(I - P^n_\beta)X = I$. Then we can solve the

following n equations with variables $x_{1,1}, ..., x_{n,1}$

$$x_{1,1} - p_{\beta_i} x_{2,1} = 1, \tag{4.12}$$

$$-(1 - p_{\beta_i}) x_{i-1,1} + x_{i,1} - p_{\beta_i} x_{i+1,1} = 0,\ i = 2...n-1, \tag{4.13}$$

$$-(1 - p_{\beta_i}) x_{n-1,1} + (1 - p_{\beta_i}) x_{n,1} = 0 \tag{4.14}$$

From equation (4.14), we can derive that $x_{n,1} = x_{n-1,1}$. And plugging this result in to the $(n-1)$-th equation, i.e. where $i = n-1$, we get $-(1 - p_{\beta_i}) x_{n-2,1} + (1 - p_\beta) x_{n,1} = 0$ which means $x_{n-2,1} = x_{n,1}$. By induction, we can show that $x_{i,1} = x_{n,1}, \forall i = 1...n$. Then we can plug this in to the equation (4.12) to obtain $x_{1,1} - p_{\beta_i} x_{1,1} = 1$ and thus $x_{i,1} = \frac{1}{1-p_{\beta_i}}, \forall i = 1...n$. □

Lemma 4.2.2. *The first row of matrix* $\left(I - P^n_{\beta_i}\right)^{-1}$ *is* $\left[\frac{1}{\widetilde{p}_{\beta_i}} ... \frac{p^{j-1}_{\beta_i}}{\widetilde{p}^j_{\beta_i}} ... \frac{p^{n-1}_{\beta_i}}{\widetilde{p}^n_{\beta_i}}\right]$.

Proof. If $n = 1$ or 2, the inverse matrix $\left(I - P^n_{\beta_i}\right)^{-1}$ has already been shown in the proof above. For $n > 2$, we still have $X(I - P^n_\lambda) = I$. Then we can solve n equations with variables $x_{1,1}, ..., x_{1,n}$.

$$x_{1,1} - (1 - p_{\beta_i}) x_{1,2} = 1, \tag{4.15}$$

$$-p_{\beta_i} x_{1,i-1} + x_{1,i} - (1 - p_{\beta_i}) x_{1,i+1} = 0,\ i = 2...n-1, \tag{4.16}$$

$$-p_{\beta_i} x_{1,n-1} + (1 - p_{\beta_i}) x_{1,n} = 0. \tag{4.17}$$

From equation (4.17), we can derive that $x_{1,n} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{1,n-1}$. And plugging this result in to the $(n-1)$-th equation, i.e. where $i = n-1$, we get

$$-p_{\beta_i} x_{1,n-2} + x_{1,n-1} - (1 - p_{\beta_i}) \frac{p_{\beta_i}}{(1 - p_{\beta_i})} x_{1,n-1} = 0.$$

Rearranging this equation, we can have $-p_{\beta_i} x_{1,n-2} + (1 - p_{\beta_i}) x_{1,n-1} = 0$ resulting in $x_{1,n-1} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{1,n-2}$. By induction, we can show that $x_{1,i} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{1,i-1}, \forall i = 2...n$.

Then we can plug this in to the equation (4.15) to obtain $x_{1,1} - (1 - p_{\beta_i})\frac{p_{\beta_i}}{1-p_{\beta_i}}x_{1,1} = x_{1,1} - p_{\beta_i}x_{1,1} = 1$, which determines $x_{1,1} = \frac{p_{\beta_i}}{1-p_{\beta_i}} = \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}$ and $x_{1,i} = \frac{p_{\beta_i}^{i-1}}{\widetilde{p}_{\beta_i}^{i}} \forall i = 1...n.$ □

Lemma 4.2.3. *The inverse of matrix $I - P^n_{/A_n}$ is*

$$\begin{bmatrix} \left(I - P^n_{\beta_1}\right)^{-1} & Q^n_{\beta_2} & Q^n_{\beta_3} & \dots & Q^n_{\beta_{N-1}} \\ \mathbf{0} & \left(I - P^n_{\beta_2}\right)^{-1} & Q^n_{\beta_3} & \dots & Q^n_{\beta_{N-1}} \\ \mathbf{0} & \mathbf{0} & \left(I - P^n_{\beta_3}\right)^{-1} & \dots & Q^n_{\beta_{N-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \left(I - P^n_{\beta_{N-1}}\right)^{-1} \end{bmatrix}$$

where $Q^n_{\beta_i}$ is an $n \times n$ matrix as

$$Q^n_{\beta_i} = \begin{bmatrix} \frac{1}{\widetilde{p}_{\beta_i}} & \dots & \frac{p^{j-1}_{\beta_i}}{\widetilde{p}^{j}_{\beta_i}} & \dots & \frac{p^{n-1}_{\beta_i}}{\widetilde{p}^{n}_{\beta_i}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\widetilde{p}_{\beta_i}} & \dots & \frac{p^{j-1}_{\beta_i}}{\widetilde{p}^{j}_{\beta_i}} & \dots & \frac{p^{n-1}_{\beta_i}}{\widetilde{p}^{n}_{\beta_i}} \end{bmatrix}. \tag{4.18}$$

Proof. Based on the definition of the inverse of a matrix, we need to show

$$(I - P^n_{\beta_i})^{-1}(I - P^n_{\beta_i}) = I,\ i = 1, ..., N-1, \tag{4.19}$$

$$-(I - P^n_{\beta_{i-1}})^{-1}C^n_{\beta_{i-1}} + Q^n_{\beta_i}(I - P^n_{\beta_i}) = \mathbf{0},\ i = 2, ..., N-1, \tag{4.20}$$

$$-Q^n_{\beta_{i-1}}C^n_{\beta_{i-1}} + Q^n_{\beta_i}(I - P^n_{\beta_i}) = \mathbf{0},\ i = 3, ..., N-1. \tag{4.21}$$

Equation (4.19) is true based on the definition of an inverse matrix.

Based on Lemma 4.2.1 and the definition of $C^n_{\beta_{i-1}}$ in equation (4.8), we can compute

$$\begin{aligned}
&\left(I-P^n_{\beta_{i-1}}\right)^{-1}C^n_{\beta_{i-1}}\\
&=\begin{bmatrix}(I-P^n_{\beta_{i-1}})^{-1}_{1,1} & \dots & (I-P^n_{\beta_{i-1}})^{-1}_{1,n}\\ \vdots & \ddots & \vdots\\ (I-P^n_{\beta_{i-1}})^{-1}_{n,1} & \dots & (I-P^n_{\beta_{i-1}})^{-1}_{n,n}\end{bmatrix}\begin{bmatrix}1-p_{\beta_{i-1}} & 0 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \dots & 0\end{bmatrix}\\
&=\begin{bmatrix}\frac{1-p_{\beta_{i-1}}}{1-p_{\beta_{i-1}}} & 0 & \dots & 0\\ \vdots & \vdots & \ddots & \vdots\\ \frac{1-p_{\beta_{i-1}}}{1-p_{\beta_{i-1}}} & 0 & \dots & 0\end{bmatrix}=\begin{bmatrix}1 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \vdots\\ 1 & 0 & \dots & 0\end{bmatrix}
\end{aligned}$$

Then we can easily show

$$\begin{aligned}
&\left(I-P^n_{\beta_{i-1}}\right)^{-1}C^n_{\beta_{i-1}}\left(I-P^n_{\beta_i}\right)^{-1}\\
&=\begin{bmatrix}1 & 0 & \dots & 0\\ \vdots & \vdots & \vdots & \vdots\\ 1 & 0 & \dots & 0\end{bmatrix}\begin{bmatrix}(I-P^n_{\beta_i})^{-1}_{1,1} & \dots & (I-P^n_{\beta_i})^{-1}_{1,n}\\ \vdots & \ddots & \vdots\\ (I-P^n_{\beta_i})^{-1}_{n,1} & \dots & (I-P^n_{\beta_i})^{-1}_{n,n}\end{bmatrix}\\
&=\begin{bmatrix}(I-P^n_{\beta_i})^{-1}_{1,1} & \dots & (I-P^n_{\beta_i})^{-1}_{1,n}\\ \vdots & \ddots & \vdots\\ (I-P^n_{\beta_i})^{-1}_{1,1} & \dots & (I-P^n_{\beta_i})^{-1}_{1,n}\end{bmatrix}
\end{aligned}$$

where every row equals the first row of matrix $(I-P^n_{\beta_i})^{-1}$. Then based on Lemma 4.2.2 and the definition of $Q^n_{\beta_i}$ in equation (4.18), we have

$$\left(I-P^n_{\beta_{i-1}}\right)^{-1}C^n_{\beta_{i-1}}\left(I-P^n_{\beta_i}\right)^{-1}=Q^n_{\beta_i}.$$

Then multiplying matrix $\left(I - P^n_{\beta_i}\right)$ to both sides of the above equation, we have

$$\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} = Q^n_{\beta_i}\left(I - P^n_{\beta_i}\right).$$
$$\Rightarrow -\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} + Q^n_{\beta_i}\left(I - P^n_{\beta_i}\right) = \mathbf{0}.$$

which shows that equation (4.20) is true for $i = 2, ..., N-1$. Similarly, for equation (4.21), we have

$$Q^n_{\beta_{i-1}} C^n_{\beta_{i-1}} = \begin{bmatrix} \frac{1}{\widetilde{p}_{\beta_{i-1}}} & \cdots & \frac{p^{n-1}_{\beta_{i-1}}}{\widetilde{p}^n_{\beta_{i-1}}} \\ \vdots & \ddots & \vdots \\ \frac{1}{\widetilde{p}_{\beta_{i-1}}} & \cdots & \frac{p^{n-1}_{\beta_{i-1}}}{\widetilde{p}^n_{\beta_{i-1}}} \end{bmatrix} \begin{bmatrix} 1-p_{\beta_{i-1}} & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \ldots & 0 \end{bmatrix} = \left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} = Q^n_{\beta_i}\left(I - P^n_{\beta_i}\right).$$
$$\Rightarrow -Q^n_{\beta_{i-1}} C^n_{\beta_{i-1}} + Q^n_{\beta_i}\left(I - P^n_{\beta_i}\right) = \mathbf{0}.$$

which shows that equation (4.21) is true.

□

Same as WMA, we will take the first element in $\mathbf{h}^n_A$ as the metric to measure the adaptiveness, i.e.

$$\mathbf{h}^n_A(1) = \sum_{i=1}^{N-1} \sum_{j=1}^{n} \frac{p^{j-1}_{\beta_i}}{\widetilde{p}^j_{\beta_i}} = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \frac{1 - \left(\frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}\right)^n}{1 - \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}}. \tag{4.22}$$

For the MEA, after B_n is removed, the states left are (α, R, n). Then $P^n_{/B}$ is an n by n matrix as

$$P^n_{/B_n} = \left[P^n_\alpha\right]. \tag{4.23}$$

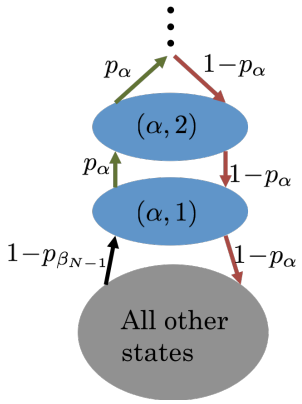

Figure 4.3: Partial Markov chain for N-Expert Winnow Algorithm. The blue circles represent the states in set A and the grey circle represents the states in set B.

For consistency, we take the first element in $\mathbf{h}_B$ as the metric. The formula is

$$\mathbf{h}_B^n(1) = \sum_{j=1}^{n} \frac{p_\alpha^{j-1}}{\widetilde{p}_\alpha^{j}}. \tag{4.24}$$

For Winnow, the set A contains states (α, R). The transition matrix is the same as $\mathbb{P}^n$ of MEA when n goes to infinity. Therefore, after A is removed, the partial transitions matrix is the limit of $P^n_{/A_n}$ as $n \to \infty$. Since $\widetilde{p}_{\beta_i} = 1 - p_{\beta_i} > 0.5 > p_{\beta_i}$, we can compute the first element in the mean hitting times as

$$\begin{aligned}\mathbf{h}_A^\infty(1) &= \lim_{n\to\infty} \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \frac{1 - \left(\frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}\right)^n}{1 - \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}} = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \frac{1}{1 - \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}} \\ &= \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i} - p_{\beta_i}}.\end{aligned} \tag{4.25}$$

For consistency, since $\widetilde{p}_\alpha = 1 - p_\alpha < 0.5 < p_\alpha$, the first element in $\mathbf{h}_B$ is

$$\mathbf{h}_B^\infty(1) = \lim_{n\to\infty} \sum_{j=1}^{n} \frac{p_\alpha^{j-1}}{\widetilde{p}_\alpha^{j}} = +\infty. \tag{4.26}$$

The fourth algorithm, HAMEA is a modified version of MEA. For each n, we define A_n to be the set that contains all the states (α, R, n) and B_n to be the set that contains all

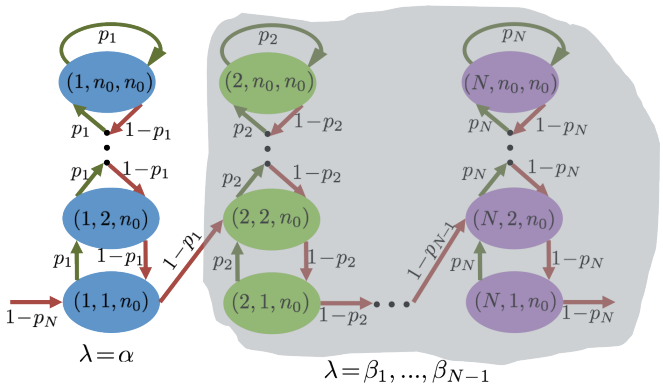

Figure 4.4: Partial Markov chains of HAMEA with $n = n_0$. The blue circles represent the states in set A_{n_0} with preferred output and the circles in the grey area represent the states in set B_{n_0}.

the states (λ, R, n) where $\lambda \neq \alpha$. The sets A_n and B_n with $n = n_0$ are illustrated by Figure 4.4. It can be easily observed that the difference between MEA and HAMEA lies only in the transitions between λ.

For this reason the partial transition matrix P_i^n for the partial Markov chain consisting of states $[i, R, n]$ for fixed i and n, is the same as the partial transition matrix of MEA given by equation 4.7. The difference lies in the matrix C_i^n. If $i < N$ and $n = 1$ then $C_i^1 = 1-p_i$. If $i = N$ and $n = 1$ then $C_N^n = \begin{bmatrix} 0 & 1-p_i \end{bmatrix}$ And for all $n > 1$.

$$C_i^n = \begin{bmatrix} 0 & 1-p_i & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{4.27}$$

Which is also an $n \times n$ matrix. Then the partial transition matrix $\mathbb{P}^n$ for the partial Markov chain consisting of states (i, R, n) for $i = 1, ..., N$ under a fixed n is the same as MEA in equation 4.9. After A_n is removed, the partial transitions matrix is shown by equation 4.10. Based on equation (4.1), to compute the mean hitting time from states in the set B_n to the set A_n, we need to compute $\left(I - P_{/A_n}^n\right)^{-1}$. In order to show this we first need to show following lemma.

Lemma 4.2.4. *The second row of matrix* $\left(I - P^n_{\beta_i}\right)^{-1}$ *is* $\left[\frac{1}{\widetilde{p}_{\beta_i}} \frac{p_{\beta_i}}{\widetilde{p}^2_{\beta_i}} ... \frac{p^{n-2}_{\beta_i}}{\widetilde{p}^n_{\beta_i}}\right]$.

Proof. If $n = 1$ or 2, the inverse matrix $\left(I - P^n_{\beta_i}\right)^{-1}$ has already been shown in proof 4.2.1. For $n > 2$, we still have $X(I - P^n_\lambda) = I$. Then we can solve n equations with variables $x_{2,1}, ..., x_{2,n}$.

$$x_{2,1} - (1 - p_{\beta_i})x_{2,2} = 0, \tag{4.28}$$

$$- p_{\beta_i}x_{2,1} + x_{2,2} - (1 - p_{\beta_i})x_{2,3} = 1 \tag{4.29}$$

$$- p_{\beta_i}x_{2,i-1} + x_{2,i} - (1 - p_{\beta_i})x_{2,i+1} = 0,\ i = 3...n - 1, \tag{4.30}$$

$$- p_{\beta_i}x_{2,n-1} + (1 - p_{\beta_i})x_{2,n} = 0. \tag{4.31}$$

From equation (4.31), we can derive that $x_{2,n} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{2,n-1}$. And plugging this result in to the $(n - 1)$-th equation, i.e. where $i = n - 1$, we get

$$-p_{\beta_i}x_{2,n-2} + x_{2,n-1} - (1 - p_{\beta_i})\frac{p_{\beta_i}}{(1 - p_{\beta_i})}x_{1,n-1} = 0.$$

Rearranging this equation, we can have $-p_{\beta_i}x_{2,n-2} + (1 - p_{\beta_i})x_{2,n-1} = 0$ resulting in $x_{2,n-1} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{2,n-2}$. By induction, we can show that $x_{2,i} = \frac{p_{\beta_i}}{1-p_{\beta_i}} x_{2,i-1}, \forall i = 3...n$. Then we can plug this in to the equation (4.29) to obtain $-p_{\beta_i}x_{2,1}+(1-p_{\beta_i})x_{2,2} = 1$. This can be combined with equation 4.28 to give $x_{2,2} = \frac{1}{(1-p_{\beta_i})^2}$. This determines $x_{2,1} = \frac{p_{\beta_i}}{1-p_{\beta_i}} = \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}$ and $x_{2,i} = \frac{p^{i-2}_{\beta_i}}{\widetilde{p}^i_{\beta_i}} \forall i = 2...n$. □

Lemma 4.2.5. *The inverse of matrix* $I - P^n_{/A_n}$ *is*

$$\begin{bmatrix} \left(I - P^n_{\beta_1}\right)^{-1} & Q^n_{\beta_2} & Q^n_{\beta_3} & \dots & Q^n_{\beta_{N-1}} \\ \mathbf{0} & \left(I - P^n_{\beta_2}\right)^{-1} & Q^n_{\beta_3} & \dots & Q^n_{\beta_{N-1}} \\ \mathbf{0} & \mathbf{0} & \left(I - P^n_{\beta_3}\right)^{-1} & \dots & Q^n_{\beta_{N-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \left(I - P^n_{\beta_{N-1}}\right)^{-1} \end{bmatrix}$$

where $Q^n_{\beta_i}$ is an $n \times n$ matrix as

$$Q^n_{\beta_i} = \begin{bmatrix} \frac{1}{\widetilde{p}_{\beta_i}} & \frac{p_{\beta_i}}{\widetilde{p}^2_{\beta_i}} & \cdots & \frac{p^{n-2}_{\beta_i}}{\widetilde{p}^n_{\beta_i}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{1}{\widetilde{p}_{\beta_i}} & \frac{p_{\beta_i}}{\widetilde{p}^2_{\beta_i}} & \cdots & \frac{p^{n-2}_{\beta_i}}{\widetilde{p}^n_{\beta_i}} \end{bmatrix}. \tag{4.32}$$

Proof. This proof follows the same method as Lemma 4.2.3. However because C^n_i is different where $n > 1$ there are slight changes to the analysis.

To show that 4.19,4.20, and 4.21 hold.

Based on Lemma 4.2.1 and the definition of $C^n_{\beta_{i-1}}$ in equation (4.27), we can compute

$$\begin{aligned} &\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} \\ &= \begin{bmatrix} (I-P^n_{\beta_{i-1}})^{-1}_{1,1} & \cdots & (I-P^n_{\beta_{i-1}})^{-1}_{1,n} \\ \vdots & \ddots & \vdots \\ (I-P^n_{\beta_{i-1}})^{-1}_{n,1} & \cdots & (I-P^n_{\beta_{i-1}})^{-1}_{n,n} \end{bmatrix} \begin{bmatrix} 0 & 1-p_{\beta_{i-1}} & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1-p_{\beta_{i-1}}}{1-p_{\beta_{i-1}}} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-p_{\beta_{i-1}}}{1-p_{\beta_{i-1}}} & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \end{aligned}$$

Then we can easily show

$$\begin{aligned}
&\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} \left(I - P^n_{\beta_i}\right)^{-1} \\
&= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} (I - P^n_{\beta_i})^{-1}_{1,1} & \dots & (I - P^n_{\beta_i})^{-1}_{1,n} \\ \vdots & \ddots & \vdots \\ (I - P^n_{\beta_i})^{-1}_{n,1} & \dots & (I - P^n_{\beta_i})^{-1}_{n,n} \end{bmatrix} \\
&= \begin{bmatrix} (I - P^n_{\beta_i})^{-1}_{2,1} & \dots & (I - P^n_{\beta_i})^{-1}_{2,n} \\ \vdots & \ddots & \vdots \\ (I - P^n_{\beta_i})^{-1}_{2,1} & \dots & (I - P^n_{\beta_i})^{-1}_{2,n} \end{bmatrix}
\end{aligned}$$

where every row equals the first row of matrix $(I - P^n_{\beta_i})^{-1}$. Then based on Lemma 4.2.4 and the definition of $Q^n_{\beta_i}$ in equation (4.32), we have

$$\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} \left(I - P^n_{\beta_i}\right)^{-1} = Q^n_{\beta_i}.$$

Then multiplying matrix $\left(I - P^n_{\beta_i}\right)$ to both sides of the above equation, we have

$$\begin{aligned}
&\left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} = Q^n_{\beta_i} \left(I - P^n_{\beta_i}\right). \\
\Rightarrow &- \left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} + Q^n_{\beta_i} \left(I - P^n_{\beta_i}\right) = \mathbf{0}.
\end{aligned}$$

which shows that equation (4.20) is true for $i = 2, ..., N-1$. Similarly, for equation (4.21),

we have

$$Q^n_{\beta_{i-1}} C^n_{\beta_{i-1}} = \begin{bmatrix} \frac{1}{\widetilde{p}_{\beta_{i-1}}} & \cdots & \frac{p^{n-2}_{\beta_{i-1}}}{\widetilde{p}^n_{\beta_{i-1}}} \\ \vdots & \ddots & \vdots \\ \frac{1}{\widetilde{p}_{\beta_{i-1}}} & \cdots & \frac{p^{n-2}_{\beta_{i-1}}}{\widetilde{p}^n_{\beta_{i-1}}} \end{bmatrix} \begin{bmatrix} 0 & 1-p_{\beta_{i-1}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 \end{bmatrix} = \left(I - P^n_{\beta_{i-1}}\right)^{-1} C^n_{\beta_{i-1}} = Q^n_{\beta_i} \left(I - P^n_{\beta_i}\right).$$

$$\Rightarrow - Q^n_{\beta_{i-1}} C^n_{\beta_{i-1}} + Q^n_{\beta_i} \left(I - P^n_{\beta_i}\right) = \mathbf{0}.$$

which shows that equation (4.21) is true. □

Same as MEA, when $n = 1$ we will take the first element in $\mathbf{h}^n_A$ as the metric to measure the adaptiveness. However when $n > 1$ in order that adaptiveness is always measured from the first reachable state in B, the second element is taken. i.e.

$$\mathbf{h}^n_A(1) = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} \frac{1 - \left(\frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}\right)}{1 - \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}}. \tag{4.33}$$

$$\mathbf{h}^n_A(2) = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} + \sum_{j=2}^{n} \frac{p^{j-2}_{\beta_i}}{\widetilde{p}^j_{\beta_i}} = \sum_{i=1}^{N-1} \frac{1}{\widetilde{p}_{\beta_i}} + \frac{1}{\widetilde{p}^2_{\beta_i}} \frac{1 - \left(\frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}\right)^{n-1}}{1 - \frac{p_{\beta_i}}{\widetilde{p}_{\beta_i}}}. \tag{4.34}$$

For the HAMEA, after B_n is removed, the states left are (α, R, n). Then $P^n_{/B}$ is an n by n matrix as

$$P^n_{/B_n} = \left[P^n_\alpha\right]. \tag{4.35}$$

For consistency, we take the first element in $\mathbf{h}_B$ if $n = 1$, and the second element if $n > 1$

as the metric. The formula is

$$\mathbf{h}_B^1(1) = \frac{1}{\widetilde{p}_\alpha}. \tag{4.36}$$

$$\mathbf{h}_B^n(2) = \frac{1}{\widetilde{p}_\alpha} + \sum_{j=2}^{n} \frac{p_\alpha^{j-2}}{\widetilde{p}_\alpha^{j}}. \tag{4.37}$$

4.2.2 Adaptiveness Analysis for Four Learning Algorithms

Adaptiveness is a measure of how fast an algorithm changes its prediction when the concept drifts, i.e. changes over time. We denote the measure of adaptiveness as t_A and we use the first element in $\mathbf{h}_A$ to quantify adaptiveness of the four learning algorithms. A smaller mean hitting steps from the initial state to the hitting set indicates better adaptiveness.

Furthermore, to make the metric for adaptiveness insensitive to the number of preferences, we will find the maximum values of these mean hitting steps under the possible values of the probabilities p_{β_i}. As a result, the metric for adaptiveness will only depend on the probability for the strong preference p_α.

WMA

For the WMA, its adaptiveness is measured by the mean hitting steps from the initial state $(\beta_1, 1)$ to the hitting set A, $\mathbf{h}_A(1)$ in equation (4.4).

Proposition 4.2.6. *Given p_α, the maximal value of $\mathbf{h}_A(1)$ for WMA is*

$$t_{A,\mathrm{WMA}} = \frac{1}{p_\alpha} + N - 2. \tag{4.38}$$

Proof. Define a vector $\mathbf{p}_\beta = [p_{\beta_1}, ..., p_{\beta_{N-1}}]^T$. Then $t_{A,\mathrm{WMA}}$ can be viewed as a function of $\mathbf{p}_\beta$. Also, the probabilities in $\mathbf{p}_\beta$ must satisfy the equality constraint $\sum_{i=1}^{N-1} p_{\beta_i} = 1 - p_\alpha$. We define the constraint function $g(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} p_{\beta_i} - (1 - p_\alpha)$. To find the maximal value

of $t_{A,\mathrm{WMA}}$, we need to solve an optimization problem

$$\begin{aligned} \max_{\mathbf{p}_\beta} \quad & t_{A,\mathrm{WMA}}(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} \frac{1}{1-p_{\beta_i}} \\ \text{s.t} \quad & g(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} p_{\beta_i} - (1-p_\alpha) = 0. \end{aligned} \tag{4.39}$$

Since the partial derivatives of $g(\mathbf{p}_\beta)$ are non-zero, we can utilize the Lagrange multiplier to find the extrema of (4.39) using the following equations

$$\frac{\partial t_{A,\mathrm{WMA}}(\mathbf{p}_\beta)}{\partial p_{\beta_i}} - c\frac{\partial g(\mathbf{p}_\beta)}{\partial p_{\beta_i}} = (1-p_{\beta_i})^{-2} - c = 0 \tag{4.40}$$

where c is the Lagrange multiplier and $i = 1, ..., N-1$. The equation (4.40) gives two solutions, $1 - \sqrt{\frac{1}{c}}$ and $1 + \sqrt{\frac{1}{c}}$. The solution $1 + \sqrt{\frac{1}{c}}$ does not satisfy $0 \leq p_{\beta_i} < 0.5$. Therefore, $p_{\beta_i} = 1 - \sqrt{\frac{1}{c}}$ for $i = 1, ..., N-1$. Then we have $p_{\beta_1} = \cdots = p_{\beta_{N-1}}$. Using the equality constraints, we can compute $p_{\beta_1} = \cdots = p_{\beta_{N-1}} = \frac{1-p_\alpha}{N-1}$. Based on this result, the only extrema is

$$t_{A,\mathrm{WMA}} = (N-1)\frac{1}{1-\frac{1-p_\alpha}{N-1}} = \frac{(N-1)^2}{N-2+p_\alpha}. \tag{4.41}$$

Next we consider the boundary conditions $p_{\beta_i} = 1 - p_\alpha$ for some $i \in \{1, ..., N-1\}$ and $p_{\beta_j} = 0, j \neq i$. Solving $t_{A,\mathrm{WMA}}$ for these boundary conditions gives

$$\begin{aligned} t_{A,\mathrm{WMA}} &= \frac{1}{1-(1-p_\alpha)} + (N-2)\frac{1}{1-0} \\ &= \frac{1}{p_\alpha} + N - 2. \end{aligned} \tag{4.42}$$

Then we need to compare the values in equation (4.41) and (4.42) by taking the difference

of them. Let

$$
\begin{aligned}
e_{\mathrm{WMA}} &= \frac{1}{p_\alpha} + N - 2 - \frac{(N-1)^2}{N-2+p_\alpha} \\
&= \frac{p_\alpha + N - 2 + p_\alpha(p_\alpha + N - 2)(N-2) - (N-1)^2 p_\alpha}{p_\alpha(p_\alpha + N - 2)} \\
&= \frac{p_\alpha + N - 2 + p_\alpha^2(N-2) + [(N-2)^2 p_\alpha - (N-1)^2 p_\alpha]}{p_\alpha(p_\alpha + N - 2)} \\
&= \frac{p_\alpha + N - 2 + p_\alpha^2(N-2) - (2N-3)p_\alpha}{p_\alpha(p_\alpha + N - 2)} \\
&= \frac{(N-2)p_\alpha^2 - (2N-4)p_\alpha + N - 2}{p_\alpha(p_\alpha + N - 2)} \\
&= \frac{(N-2)(p_\alpha^2 - 2p_\alpha + 1)}{p_\alpha(p_\alpha + N - 2)} = \frac{(N-2)(p_\alpha - 1)^2}{p_\alpha(p_\alpha + N - 2)}
\end{aligned} \tag{4.43}
$$

Since $N \geq 2$ and $0 < p_\alpha \leq 1$, we have that $(N-2)(p_\alpha - 1)^2 \geq 0$ and $p_\alpha(p_\alpha + N - 2) > 0$. Therefore, $e_{\mathrm{WMA}} \geq 0$. This means that $t_{A,\mathrm{WMA}}$ at the boundary is always greater than or equal to $t_{A,\mathrm{WMA}}$ at the extrema. Thus $\frac{1}{p_\alpha} + N - 2$ is the maximal value. □

Winnow

For the Winnow, its adaptiveness is measured by the mean hitting steps from state $(\beta_1, 1)$ to hitting set A, $\mathbf{h}_A(1)$ in equation (4.25).

Proposition 4.2.7. *Given p_α, the maximum value of $\mathbf{h}_A(1)$ for Winnow is*

$$
t_{A,\mathrm{WIN}} = \frac{1}{p_\alpha - \widetilde{p}_\alpha} + N - 2. \tag{4.44}
$$

Proof. Similarly as the proof for Proposition 4.2.6, to find the maximal value of $t_{A,\mathrm{WIN}}$, we need to solve an optimization problem

$$
\begin{aligned}
&\max_{\mathbf{p}_\beta} \quad t_{A,\mathrm{WIN}}(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} \frac{1}{1 - 2p_{\beta_i}} \\
&\text{s.t} \quad g(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} p_{\beta_i} - (1 - p_\alpha) = 0.
\end{aligned} \tag{4.45}
$$

we can utilize the Lagrange multiplier to find the extrema of (4.45) using

$$\frac{\partial t_{A,\text{WIN}}(\mathbf{p}_\beta)}{\partial p_{\beta_i}} - c\frac{\partial g(\mathbf{p}_\beta)}{\partial p_{\beta_i}} = 2\left(1-p_{\beta_i}\right)^{-2} - c = 0 \tag{4.46}$$

where c is the Lagrange multiplier and $i = 1, ..., N-1$. The equation (4.46) gives two solutions, $1-\sqrt{\frac{2}{c}}$ and $1+\sqrt{\frac{2}{c}}$. The solution $1+\sqrt{\frac{2}{c}}$ does not satisfy $0 \leq p_{\beta_i} < 0.5$. Therefore, $p_{\beta_i} = 1-\sqrt{\frac{2}{c}}$ for $i = 1, ..., N-1$. Then we have $p_{\beta_1} = \cdots = p_{\beta_{N-1}}$. Using the equality constraints, we can compute $p_{\beta_1} = \cdots = p_{\beta_{N-1}} = \frac{1-p_\alpha}{N-1}$. Based on this result, the only extrema is

$$t_{A,\text{WIN}} = (N-1)\frac{1}{1-2\frac{1-p_\alpha}{N-1}} = \frac{(N-1)^2}{N-3+2p_\alpha}. \tag{4.47}$$

For the boundary conditions $p_{\beta_i} = 1-p_\alpha$ and $p_{\beta_j} = 0, j \neq i$, we have

$$t_{A,\text{WIN}} = \frac{1}{1-2(1-p_\alpha)} + N - 2 = \frac{1}{2p_\alpha - 1} + N - 2. \tag{4.48}$$

Taking the difference of (4.47) and (4.48), we have

$$\begin{aligned}
e_{\text{WIN}} &= \frac{1}{2p_\alpha-1} + N - 2 - \frac{(N-1)^2}{N-3+2p_\alpha} \\
&= \frac{2p_\alpha+N-3+(2p_\alpha-1)[(2p_\alpha+N-3)(N-2)-(N-1)^2]}{(2p_\alpha-1)(2p_\alpha+N-3)} \\
&= \frac{2p_\alpha+N-3+(2p_\alpha-1)[(2p_\alpha(N-2)-(3N-5)]}{(2p_\alpha-1)(2p_\alpha+N-3)} \\
&= \frac{4(N-2)p_\alpha^2-2(4N-8)p_\alpha+4N-8}{(2p_\alpha-1)(2p_\alpha+N-3)} \\
&= \frac{4(N-2)(p_\alpha^2-2p_\alpha+1)}{(2p_\alpha-1)(2p_\alpha+N-3)} = \frac{4(N-2)(p_\alpha-1)^2}{(2p_\alpha-1)(2p_\alpha+N-3)}
\end{aligned} \tag{4.49}$$

Since $N \geq 2$ and $0.5 < p_\alpha \leq 1$, we have $(p_\alpha-1)^2 \geq 0$, $p_\alpha - 0.5 > 0$, $2p_\alpha - 1 > 0$ and $2p_\alpha + N - 3 > 0$. Therefore, $e_{\text{WIN}} \geq 0$. This means that $t_{A,\text{WIN}}$ at the boundary is always greater than or equal to $t_{A,\text{WIN}}$ at the extrema. Thus $\frac{1}{2p_\alpha-1} + N - 2$ is the maximal value

for $t_{A,\text{WIN}}$. □

MEA

For the MEA, its adaptiveness is measured by the mean hitting steps from the state $(\beta_1, 1, n)$ to hitting set A_n, i.e. $\mathbf{h}_A^n(1)$, based on equation (4.22).

Proposition 4.2.8. *Given p_α, the maximum value of $\mathbf{h}_A^n(1)$ is*

$$t_{A,\text{MEA}}^n = \sum_{i=1}^{n} \left[\frac{(\widetilde{p}_\alpha)^{i-1}}{p_\alpha^i} \right] + N - 2. \tag{4.50}$$

Proof. To find the maximal value of $t_{A,\text{MEA}}^n$, we need to solve an optimization problem

$$\begin{aligned} \max_{\mathbf{p}_\beta} \quad & t_{A,\text{MEA}}^n(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} \sum_{j=1}^{n} \frac{p_{\beta_i}^{j-1}}{\widetilde{p}_{\beta_i}^{j}} \\ \text{s.t} \quad & g(\mathbf{p}_\beta) = \sum_{i=1}^{N-1} p_{\beta_i} - (1 - p_\alpha) = 0. \end{aligned} \tag{4.51}$$

We utilize the Lagrange multiplier to find the extrema of (4.51) using

$$\frac{\partial t_{A,\text{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_i}} - c \frac{\partial g(\mathbf{p}_\beta)}{\partial p_{\beta_i}} = 0$$

Compute the first partial derivative and we can get

$$\begin{aligned} \frac{\partial t_{A,\text{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_i}} &= \sum_{j=1}^{n} \frac{\partial \left(\frac{p_{\beta_i}^{j-1}}{\widetilde{p}_{\beta_i}^{j}} \right)}{\partial p_{\beta_i}} \\ &= \sum_{j=1}^{n} \frac{(p_{\beta_i} + j - 1)\, p_{\beta_i}^{j-2}}{(1 - p_{\beta_i})^{j+1}} = c. \end{aligned} \tag{4.52}$$

Since $0 \leq p_{\beta_i} < 0.5$ for $i = 1...N-1$, we can conclude that $p_{\beta_1} = ... = p_{\beta_{N-1}}$ if equation (4.52) holds for all i. We can prove it by contradiction. If we assume that there exist a pair of indices, k and q, such that $0 \leq p_{\beta_q} < p_{\beta_k} < 0.5$ and $\frac{\partial t_{A,\text{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_q}} = \frac{\partial t_{A,\text{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_k}} = c$, then

we have $p_{\beta_q}+j-1<p_{\beta_k}+j-1$, $p_{\beta_q}^j \leq p_{\beta_k}^j$ and $\left(1-p_{\beta_q}\right)^{j+1}>\left(1-p_{\beta_k}\right)^{j+1}$. Therefore, the j-th terms in equation (4.52) of indices q and k satisfy

$$\frac{\left(p_{\beta_q}+j-1\right)p_{\beta_q}^{j-2}}{\left(1-p_{\beta_q}\right)^{j+1}}<\frac{\left(p_{\beta_k}+j-1\right)p_{\beta_k}^{j-2}}{\left(1-p_{\beta_k}\right)^{j+1}}\ \forall j=1,...,n.$$

The sum of every term results in

$$\sum_{j=1}^{n}\frac{\left(p_{\beta_q}+j-1\right)p_{\beta_q}^{j-2}}{\left(1-p_{\beta_q}\right)^{j+1}}<\sum_{j=1}^{n}\frac{\left(p_{\beta_k}+j-1\right)p_{\beta_k}^{j-2}}{\left(1-p_{\beta_k}\right)^{j+1}}.$$

This means that $\frac{\partial t_{A,\mathrm{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_q}}<\frac{\partial t_{A,\mathrm{MEA}}^n(\mathbf{p}_\beta)}{\partial p_{\beta_k}}$, which contradicts to our assumption. Then we have shown that $p_{\beta_1}=...=p_{\beta_{N-1}}$. Using the equality constraints, we can compute $p_{\beta_1}=\cdots=p_{\beta_{N-1}}=\frac{1-p_\alpha}{N-1}$. Based on this result, the only extrema is

$$\begin{aligned}t_{A,\mathrm{MEA}}^n&=(N-1)\sum_{j=1}^{n}\frac{\frac{1-p_\alpha}{N-1}^{j-1}}{\left(1-\frac{1-p_\alpha}{N-1}\right)^j}\\&=\sum_{j=1}^{n}\frac{(1-p_\alpha)^{j-1}(N-1)^2}{(p_\alpha+N-2)^j}.\end{aligned}\tag{4.53}$$

For the boundary conditions $p_{\beta_i}=1-p_\alpha$ and $p_{\beta_j}=0, j\neq i$, we have

$$\begin{aligned}t_{A,\mathrm{MEA}}^n&=\sum_{j=1}^{n}\frac{(1-p_\alpha)^{j-1}}{(1-1+p_\alpha)^j}+\sum_{k=1,k\neq i}^{N-1}\sum_{j=1}^{n}\frac{0^{j-1}}{(1-0)^j}\\&=\sum_{j=1}^{n}\left[\frac{(1-p_\alpha)^{j-1}}{p_\alpha^j}\right]+N-2.\end{aligned}\tag{4.54}$$

Taking the difference of (4.53) and (4.54), we have

$$e_{\mathrm{MEA}}=\sum_{j=1}^{n}\left[\frac{(1-p_\alpha)^{j-1}}{p_\alpha^j}\right]+N-2-\sum_{j=1}^{n}\frac{(1-p_\alpha)^{j-1}(N-1)^2}{(p_\alpha+N-2)^j}.$$

Separating the first terms of the two sums, we can rewrite the above equation as

$$e_{\mathrm{MEA}} = \frac{1}{p_\alpha} + \sum_{j=2}^{n}\left[\frac{(1-p_\alpha)^{j-1}}{p_\alpha^j}\right] + N - 2 - \frac{(N-1)^2}{p_\alpha+N-2} - \sum_{j=2}^{n}\frac{(1-p_\alpha)^{j-1}(N-1)^2}{(p_\alpha+N-2)^j}.$$

Then we can rearrange the equation as

$$e_{\mathrm{MEA}} = \left[\frac{1}{p_\alpha} + N - 2 - \frac{(N-1)^2}{p_\alpha+N-2}\right] + \sum_{j=1}^{n}\left[\frac{(1-p_\alpha)^j}{p_\alpha^{j+1}} - \frac{(1-p_\alpha)^j(N-1)^2}{(p_\alpha+N-2)^{j+1}}\right]. \tag{4.55}$$

The first term $\frac{1}{p_\alpha}+N-2-\frac{(N-1)^2}{p_\alpha+N-2} \geq 0$ has already be shown in Proposition 4.2.6. Then we need to show the second term is also greater than or equal to 0. For the j-th term of the sum, we have

$$\frac{(1-p_\alpha)^j}{p_\alpha^{j+1}} - \frac{(1-p_\alpha)^j(N-1)^2}{(p_\alpha+N-2)^{j+1}} = \frac{1-p_\alpha}{p_\alpha^2}\left(\frac{1-p_\alpha}{p_\alpha}\right)^{j-1} - \frac{(1-p_\alpha)(N-1)^2}{(p_\alpha+N-2)^2}\left(\frac{1-p_\alpha}{p_\alpha+N-2}\right)^{j-1}.$$

Since $N \geq 2$ and $1 \geq p_\alpha > 0.5$, we can derive $p_\alpha + N - 2 \geq p_\alpha$, which leads to

$$\frac{1}{p_\alpha} \geq \frac{1}{p_\alpha+N-2} \Rightarrow \left(\frac{1-p_\alpha}{p_\alpha}\right)^{j-1} \geq \left(\frac{1-p_\alpha}{p_\alpha+N-2}\right)^{j-1}. \tag{4.56}$$

To show $e_{\text{MEA}} \geq 0$, we just need to show the coefficient $\frac{1-p_\alpha}{p_\alpha^2} \geq \frac{(1-p_\alpha)(N-1)^2}{(p_\alpha+N-2)^2}$ by computing

$$\begin{aligned}
&\frac{1-p_\alpha}{p_\alpha^2} - \frac{(1-p_\alpha)(N-1)^2}{(p_\alpha+N-2)^2} = (1-p_\alpha)\frac{(p_\alpha+N-2)^2 - (N-1)^2 p_\alpha^2}{p_\alpha^2\,(p_\alpha+N-2)^2} \\
&= (1-p_\alpha)\frac{(p_\alpha+N-2)^2 - [(N-1)p_\alpha]^2}{p_\alpha^2\,(p_\alpha+N-2)^2} \\
&= \frac{(Np_\alpha + N - 2)\,(N-2)(1-p_\alpha)^2}{p_\alpha^2\,(p_\alpha+N-2)^2}.
\end{aligned}$$

Since $(1-p_\alpha)^2 \geq 0$, $N-2 \geq 0$, and $Np_\alpha + N - 2 \geq 0$, we have

$$\frac{1-p_\alpha}{p_\alpha^2} \geq \frac{(1-p_\alpha)(N-1)^2}{(p_\alpha+N-2)^2} \geq 0. \tag{4.57}$$

Combining equations (4.55), (4.56) and (4.57), we can show that $e_{\text{MEA}} \geq 0$. This means that $t^n_{A,\text{MEA}}$ at the boundary is greater than or equal to $t^n_{A,\text{MEA}}$ at the extrema. Thus $t^n_{A,\text{MEA}} = \sum_{i=1}^{n}\left(\frac{\widetilde{p}_\alpha^{\,i-1}}{p_\alpha^i}\right) + N - 2$ is the maximal value.

□

HAMEA

For the HAMEA, its adaptiveness is measured by the mean hitting steps from the state $(\beta_1, 1, 1)$ when $n = 1$ and $(\beta_1, 2, n)$ when $n > 1$ to hitting set A_n, i.e. $\mathbf{h}^1_A(1)$ or $\mathbf{h}^n_A(2)$, based on equation (4.33).

Proposition 4.2.9. *Given p_α, the maximum value of $\mathbf{h}^n_A(2)$ is*

$$t^n_{A,\text{HAMEA}} = \frac{1}{\alpha} + N - 2 + \sum_{i=2}^{n}\left(\frac{(1-\alpha)^{i-2}}{\alpha^i} + \frac{N-2}{n}\right). \tag{4.58}$$

Proof. We utilize the Lagrange multiplier to find the extrema of the system using the following set of equations.

$$\nabla t_{A,\text{HAMEA}} = c\nabla g(\beta)$$

$$g(\beta) = 1 - \alpha$$

Thus $\frac{1}{c} = \frac{1}{1-\beta_1} + \sum_{i=2}^{n} \frac{\beta_1^{i-2}}{(1-\beta_1)^i} = ... = \frac{1}{1-\beta_{N-1}} + \sum_{i=2}^{n} \frac{\beta_{N-1}^{i-2}}{(1-\beta_{N-1})^i}$. Since $0 \leq \beta_i < 0.5 \forall i = 1...N-1$ The only time this equation is true is when $\beta_1 = ... = \beta_{N-1}$

Since $g(\beta) = 1-\alpha = \sum_{i=1}^{N-1} \beta_i$, and $\beta_1 = ... = \beta_{N-1}$ Thus $1-\alpha = N-1\beta_1$, making $\beta_1 = ... = \beta_{N-1} = \frac{1-\alpha}{N-1}$ Therefore the only extrema is $t_{A,\text{HAMEA}} = \frac{(N-1)^2}{N-2+\alpha} + (N-1) \sum_{i=2}^{n} \frac{\frac{1-\alpha}{N-1}^{i-2}}{(1-\frac{1-\alpha}{N-1})^i}$.

Next we consider the boundary conditions $\beta_i = 1 - \alpha$ and $\beta_j = 0 \forall j \neq i$. Solving $t_{A,\text{HAMEA}}$ for these boundary conditions yields $t_{A,\text{HAMEA}} = \frac{1}{\alpha} + \sum_{i=2}^{n} \frac{(1-\alpha)^{i-2}}{(1-1+\alpha)^i} + \sum_{j=2}^{N} \frac{1}{1} + \sum_{i=1}^{n} \frac{0^{i-1}}{(1-0)^i} = \frac{1}{\alpha} + N - 2 \sum_{i=1}^{n} \frac{(1-\alpha)^{i-1}}{(\alpha)^i} + \frac{N-2}{n}$

Since $N \geq 2$ and $1 \leq \alpha > 0.5$. $(1-\alpha) \geq 0$, $N-2 \geq 0$, $\alpha N + N - 2 \geq 0$, $\alpha^2 \geq 0$, and $(\alpha + N - 2)^2 \geq 0$. And combining all these equations we have

$$\frac{(1-\alpha)(N-2)(\alpha N + N - 2)}{\alpha^2(\alpha + N - 2)^2} \geq 0$$

$$\frac{(1-\alpha)(N-2)(\alpha N + N - 2)}{\alpha^2(\alpha + N - 2)^2} + \frac{(N-1)^2}{(\alpha + N - 2)^2} \geq \frac{(N-1)^2}{(\alpha + N - 2)^2}$$

$$\frac{(\alpha + N - 2)^2}{\alpha^2(\alpha + N - 2)^2} \geq \frac{(N-1)^2}{(\alpha + N - 2)^2}$$

$$\frac{1}{\alpha^2} \geq \frac{(N-1)^2}{(\alpha + N - 2)^2}$$

In addition, because $N - 2 \geq 0$.

$$N - 2 \geq 0$$

$$N - 2 + \alpha \geq \alpha$$

$$\frac{1}{\alpha} \geq \frac{1}{N - 2 + \alpha}$$

$$\frac{1-\alpha}{\alpha} \geq \frac{1-\alpha}{N - 2 + \alpha}$$

Combining these two results it is apparent that.

$$\frac{1}{\alpha^2}\left(\frac{1-\alpha}{\alpha}\right)^j \geq \frac{(N-1)^2}{(\alpha+N-2)^2}\left(\frac{1-\alpha}{N-2+\alpha}\right)^j \forall j \geq 0$$

$$\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}}{\alpha^i} \geq \sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}(N-1)^2}{(\alpha+N-2)^i}$$

From what was shown in Proposition 4.2.6 we see that

$$\frac{1}{\alpha}+N-2+\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}}{\alpha^i} \geq \frac{(N-1)^2}{\alpha+N-2}+\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}(N-1)^2}{(\alpha+N-2)^i}$$

$$\frac{1}{\alpha}+N-2+\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}}{\alpha^i}+\frac{N-2}{n} \geq \frac{(N-1)^2}{\alpha+N-2}+\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}(N-1)^2}{(\alpha+N-2)^i}$$

This means that $t_{A,\text{HAMEA}}$ at the boundary is greater than or equal to $t_{A,\text{HAMEA}}$ at the extrema. Thus $\frac{(N-1)^2}{\alpha+N-2}+\sum_{i=2}^{n}\frac{(1-\alpha)^{i-2}(N-1)^2}{(\alpha+N-2)^i}$ is the minimum value. And $t^n_{A,\text{HAMEA}} = \frac{1}{\alpha}+N-2+\sum_{i=2}^{n}\left(\frac{(1-\alpha)^{i-2}}{\alpha^i}+\frac{N-2}{n}\right)$ is the maximum value. □

4.2.3 Consistency Analysis for Four Learning Algorithms

Consistency is a measure of how often a learning algorithm changes its prediction in case that there is a temporary change of the concept preference. We denote the measure of consistency as t_C and we use the first element in $\mathbf{h}_B$ to quantify consistency of WMA, Winnow, and MEA. For HAMEA, as with Adaptiveness we use the second element. A larger mean hitting time indicates a better consistency.

WMA

For the WMA, its consistency is measured by the mean hitting time from the initial state $(\alpha, 1)$ to the hitting set B. Based on equation (4.6), the formula to compute consistency for

WMA is

$$t_{C,\mathrm{WMA}} = \frac{1}{\widetilde{p}_\alpha}. \tag{4.59}$$

Winnow

For the Winnow, its consistency is measured by the mean hitting time from the initial state $(\alpha, 1)$ to the hitting set B. Based on equation (4.26), the formula to compute consistency for Winnow is

$$t_{C,\mathrm{WIN}} = +\infty. \tag{4.60}$$

MEA

For MEA, its consistency is measured by the mean hitting time from the initial state $(\alpha, 1, n)$ to the hitting set B_n. Based on equation (4.24), the formula to compute consistency for MEA is

$$t^n_{C,\mathrm{MEA}} = \sum_{i=1}^{n} \frac{p_\alpha^{i-1}}{\widetilde{p}_\alpha^{i}} = \frac{1}{\widetilde{p}_\alpha} \frac{1 - \left(\frac{p_\alpha}{\widetilde{p}_\alpha}\right)^n}{1 - \frac{p_\alpha}{\widetilde{p}_\alpha}}. \tag{4.61}$$

HAMEA

For HAMEA, its consistency is measured by the mean hitting time from the initial state $(\alpha, 1, 1)$ if $n = 1$ and $(\alpha, 2, n)$ if $n > 1$ to the hitting set B_n. Based on equation (4.36), the formula to compute consistency for HAMEA is

$$t^n_{C,\mathrm{HAMEA}} = \frac{1}{\widetilde{p}_\alpha} + \sum_{i=2}^{n} \frac{p_\alpha^{i-2}}{\widetilde{p}_\alpha^{i}} = \frac{1}{\widetilde{p}_\alpha} + \frac{1}{\widetilde{p}_\alpha^2} \frac{1 - \left(\frac{p_\alpha}{\widetilde{p}_\alpha}\right)^{n-1}}{1 - \frac{p_\alpha}{\widetilde{p}_\alpha}}. \tag{4.62}$$

4.2.4 Comparisons among the Four Learning Algorithms

When a strong preference exists and $n = 1$,

$$t^1_{C,\mathrm{MEA}} = t^1_{C,\mathrm{HAMEA}} = \frac{1}{\widetilde{p}_\alpha} \text{ and } t^1_{A,\mathrm{MEA}} = t^1_{A,\mathrm{HAMEA}} = \frac{1}{p_\alpha} + N - 2. \tag{4.63}$$

These equations are identical to equations (4.59) and (4.38) that give the consistency and adaptiveness of WMA.

When $n \to \infty$, since $p_\alpha > \widetilde{p}_\alpha$, the formula for an infinite sum of a geometric sequence gives

$$t^\infty_{C,\mathrm{MEA}} = t^\infty_{C,\mathrm{HAMEA}} = +\infty \tag{4.64}$$

and

$$t^\infty_{A,\mathrm{MEA}} = \frac{1}{p_\alpha - \widetilde{p}_\alpha} + N - 2. \tag{4.65}$$

and

$$t^\infty_{A,\mathrm{HAMEA}} = \frac{2}{p_\alpha - \widetilde{p}_\alpha} + 2N - 4. \tag{4.66}$$

For MEA these equations are identical to equations (4.60) and (4.44) that give the consistency and adaptiveness of Winnow. Hence, the WMA and the Winnow algorithms can be seen as extreme cases of the MEA. For HAMEA the consistency is the same as Winnow at infinity, however the value for adaptiveness is larger than MEA and Winnow, meaning that HAMEA is less adaptive.

For all the finite nonzero values of n, $t^n_{A,\mathrm{MEA}}$ and $t^n_{C,\mathrm{MEA}}$ take values between the two extreme values. This has confirmed our observations from prior simulations and experiments that the MEA is more adaptive than the WMA, and more consistent than the Winnow algo-

rithm [42]. As the number of times that MEA changed output increases, corresponding to an increase in n, MEA becomes less adaptive and more consistent.

For all the finite values of $n > 1$, $t^n_{A,\mathrm{HAMEA}}$ and $t^n_{C,\mathrm{HAMEA}}$ are larger than $t^n_{A,\mathrm{MEA}}$and $t^n_{C,\mathrm{MEA}}$. Thus HAMEA is more consistent but less adaptive than MEA. And as with MEA, as the number of times that HAMEA changed output increases, corresponding to an increase in n, HAMEA becomes less adaptive and more consistent.

4.3 Simulation Results

In this section, we simulate WMA, Winnow, MEA and HAMEA to learn a concept. The simulation results support the analysis we presented in the previous Section.

We assume the preferred action of the concept corresponds to the prediction of expert 1, which means $\alpha = 1$ in the simulation setup. Based on the initial weights of these learning algorithms, the initially selected expert is always expert 1 which matches with the concept the algorithms want to learn. Here we note that the variable n introduced for the MEA has the physical meaning as the number of times that a learning algorithm continuously predicted the preferred action 1. We will use this definition of n for the comparison among WMA, Winnow, MEA and HAMEA.

4.3.1 Consistency

We simulated WMA, MEA and HAMEA, each learning algorithm with 2, 3, and 7 experts respectively. We set the probability for the preferred action, p_α, to be 0.7. That is to say, the concept has a deviation probability of 0.3. Under this deviation, we want to test whether each of these learning algorithm can consistently select expert 1. For each algorithm, We ran 400 trials for each algorithm. The simulation of a trial terminates when the value of n is greater than 6. The total time steps of all trials we ran for WMA and MEA are no more than 10000. We define the number of the time steps from the initial time to the last time before the n-th prediction switching from the action 1 to other action as t^i_n, where i

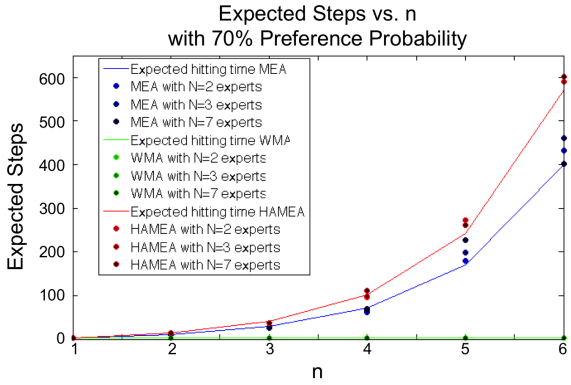

Figure 4.5: Expected and averaged steps until switching for a given n with 70% preference probability

represents the number of trials and $n = 1, ..., 6$. For example, if the prediction output of a trial i is $[1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, ...]$, then the measured time steps t_n^i are

$$[\underbrace{1, 1, 1, 1}_{t_1^i=4}, 2, 2, \underbrace{1, 1}_{t_2^i=2}, 2, 2, \underbrace{1, 1, 1, 1, 1}_{t_3^i=5}, ...].$$

We then compute the averaged time steps t_n among 400 trials for each n, where $t_n = \frac{1}{400} \sum_{i=1}^{400} t_n^i$. The larger averaged number t_n means better consistency. Figure 4.5 presents the simulation results. As we can see from this figure, WMA has a smaller averaged number, meaning that WMA is less consistent. The average numbers of MEA are larger than WMA, which means that MEA has better consistency than WMA, and the average number of HAMEA are larger than MEA, which means that HAMEA has better consistency than MEA. We can also observe that an exponential increase occurs in the number of steps as the value of n increases. This also justifies our conclusion in Section 4.2.4 that MEA and HAMEA become more consistent as n increases. Notice that Figure 4.5 does not include the Winnow Algorithm. This is because the averaged time steps of Winnow is greater than 10000, which is significantly larger than the averaged time steps for WMA, MEA and HAMEA. Therefore, the results for Winnow are not included in this figure but are verified to be greater than the averaged time steps of MEA in our simulation. This verifies that

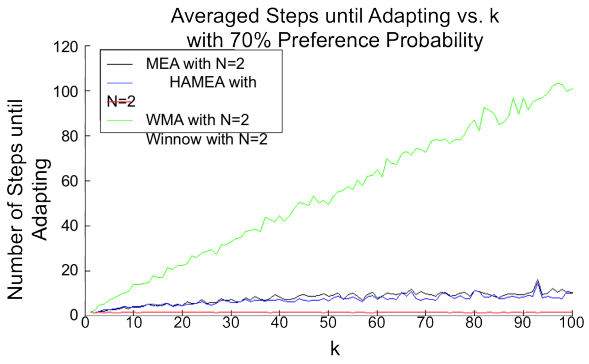

Figure 4.6: Averaged steps until adapting for a given number of steps before drift with 70% preference probability of two expert algorithms.

Winnow is more consistent than MEA and HAMEA, which has already been discussed in our analysis.

4.3.2 Adaptiveness

We simulated the case where the preferred action of the concept changed, i.e. $\alpha \neq 1$, after several time steps. The number of time steps before drift takes place, denoted by k, was ranged from 1 to 100 and 100 trials were simulated for each k. The deviation probability was set to be 0.3 both before, and after drift. For each trial, we counted how many time steps are needed for a learning algorithm to change its prediction from 1 to the new preferred action. Then we take the average of these time steps among the 100 trials for each k.

Figure 4.6 shows the average time steps of WMA, Winnow, MEA and HAMEA, each with two experts. The reason why only two experts are considered in this simulation is because this analysis focuses on measuring how long each type of the learning algorithm needs to leave a non-preferred state. Therefore, the case with 2 experts is enough to support the analysis.

As we can see from Figure 4.6, for WMA, MEA and HAMEA, the averaged time steps required to adapt to the new preference remains almost a constant over all k. The Winnow has a linear increase in the average time steps required to adapt as k increases. For WMA,

the growth is slow because it has a small $t_{A,\mathrm{WMA}}$ and the initial distance to the switch state is always 1. For MEA and HAMEA, the growth is also slow, but faster than WMA. This is because $t_{A,\mathrm{MEA}}$ and $t_{A,\mathrm{HAMEA}}$ are limited by n which, as shown in figure 4.5, becomes less likely to change after each increase in n. The value of n also limits the initial distance to the switch state. For the Winnow algorithm, the growth is much faster because it not only has a larger $t_{A,\mathrm{WIN}}$, but its initial distance to the switch state increases when the number of trials increase before the drift occurs. This shows that as the run-time of a learning algorithm increases, the Winnow algorithm becomes less adaptive and therefore it is not ideal for long-term learning. It should also be noted that HAMEA has a slightly smaller adaptivness value than MEA. This is because of the increased consistency that is shown in the previous subsection allows HAMEA to remain at smaller n than MEA. Thus HAMEA and MEA have better consistency than WMA, and better adaptiveness than Winnow.

4.4 Experimental Results

4.4.1 Setup

The experiment consisted of a simple hallway with a stationary 'robot' in the center. Ten human participants were then asked to walk past the stationary robot at least ten times. The direction the human chose when passing this robot was recorded. Then, using the assumption that the human would prefer the robot to pass in such a way so that there would not be a collision, the expected output of the robot was determined. Thus the opposite side of whatever side the human used to pass the stationary robot can be taken to be the 'success' output.

This stationary robot was equipped with a distance sensor to determine the direction that the human passed the robot. And a data set was compiled for each person giving the sequential results for the expected output of the robot. WMA, Winnow, MEA, and HADEA were then run on these data sets. The human expectation and algorithm outputs can be seen in appendix A, as well as the recorded Error and n as defined in the previous section.

4.4.2 Results

Consistency can be related to n for an algorithm. As the larger the value that n reaches in a set number of steps the more times the algorithm has changed outputs and thus the the algorithm performance is less consistent. From the human experimental results WMA had an average n of 4.1, Winnow had an average n of 1.7, MEA had an average n of 3.4, and HAMEA had an average n of 2.1. This confirms both analysis and simulation with WMA being the least consistent, followed by MEA, then HAMEA then Winnow as the most consistent. This pattern holds for each individual test except for subject 7 where HAMEA behaved more consistently than Winnow.

The Markov Chain model can be leveraged to calculate p_1 using two methods that are explained in appendix B. These two methods are called Full State (FS) which utilizes every recorded state, and First State Last State (FSLS) which only required the initial and final state of the learning algorithm to be recorded.

FS Analysis

Since the calculation for FS means that all the transitions are considered with equal weight. The final result is the same as if each output was recorded and calculated. This final result will be included in the FSLS Analysis as the expected p_1.

p_1 Calculation

As explained in appendix B FS analysis will return the same p_1 value for every algorithm when $N = 2$. This value is also the same as the value for p_1 obtained by directly using the collected human data.

The probability output produced by FSLS analysis can be seen in figure 4.7. This figure shows the results of the MEA algorithm run on the data-set of subject 1. It gives the probably that each potential p_1 would result in the known final state which shows the probability relating to each p_1. The largest value is $p_1 = .85$ which is close to actual

Table 4.1: Table of results for FSLS analysis on experimental data

		WMA		Winnow		MEA		HAMEA	
Subject	Expected p_1	p_1	error	p_1	error	p_1	error	p_1	error
1	.8462	1.0	.1538	.85	.0038	.85	.0038	.85	.0038
2	.4167	0.0	.4167	.40	.0167	0.0	.4167	0.0	.4167
3	.8333	1.0	.1667	.85	.0167	.85	.0167	.85	.0167
4	.3333	1.0	.6667	.35	.0167	.35	.0167	.35	.0167
5	.5000	0.0	.5000	.50	0.000	.45	.0500	.45	.0500
6	.5385	0.0	.5385	.55	.0115	.45	.0715	.45	.0715
7	.7143	1.0	.2857	.70	.0143	.65	.0643	.65	.0643
8	.3333	0.0	.3333	.35	.0167	.35	.0167	.35	.0167
9	.5000	0.0	.5000	.50	0.000	.45	.0500	.45	.0500
10	.5000	0.0	.5000	.50	0.000	.55	.0500	.55	.0500
Average Error		.5562		.0096		.0756		.0756	

expected value of $p_1 = .8462$

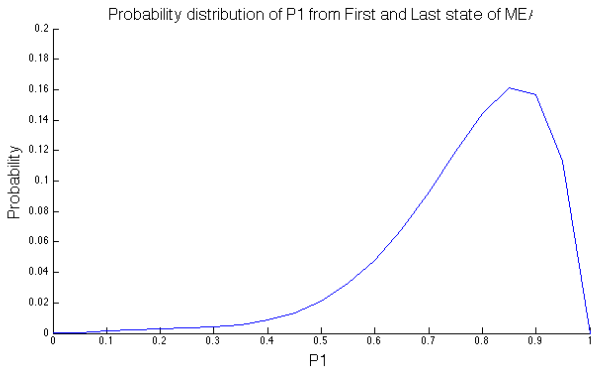

Figure 4.7: Subject 1 FSLS parameter ID with MEA algorithm.

Table 4.1 shows the results of FSLS analysis for ten data sets as well as the expected value and the error between the expected and actual value. It can be seen that this method gives the most accurate results to Winnow because every step changes the state. MEA and HAMEA also have a good accuracy, although it is not as low as Winnow. WMA is the least accurate since there are only two states in the Markov chain.

CHAPTER 5

CO-LEARNING

5.1 Problem Setup

In previous chapters, we assumed that the expectation is only affected by drift and independent of the performance of the learning algorithm used by the robot. This assumption does not always hold. Perhaps the most obvious example would be when the robot is interacting with an individual human. Because the human is also able to learn, the human's expectation of the robot can change due to the changes in robot's behavior. Therefore, we study the mutual influence between the changing expectation of a human and the learning algorithm used by a robot. In this chapter, we only consider a simple case where both the human and the robot have only two possible actions, but it will illustrate how to analyze when chatter occurs and how to solve the chatter issue during co-learning. The actions for human is denoted by A_t, representing the action taken by the human at t-th interaction. It can have the value of -1 or 1. The two possible actions for the robot are denoted by λ_t, representing the action taken by the robot at t-th interaction, which also has the value of -1 or 1. If $A_t = \lambda_t$, then we say the human action and the robot action matches. Otherwise, the two actions do not match.

We model a co-learning system using two feedback loops, as shown in Figure 5.1. For each human robot interaction, the human has an expectation of what action the robot will take for the upcoming interaction. Based on the expectation, the human takes one corresponding action out of two possible human actions. Feedback is given to the human by comparing the robot's action to the human's action. If they match with each other, then the feedback is a success. Otherwise, the feedback is an error. Based on the feedback, we assume the human will adjust its internal states (characterized by some internal parameters)

and may change the expectation for the next interaction. Similar to the learning algorithm, the algorithm predicts an output based on its learning about the human. The output determines one robot behavior out of two. Feedback is given to the learning algorithm after an interaction to adjust the internal parameters of learning algorithm, in order to make a correct prediction for the next interaction. The whole co-learning system can be viewed as a closed loop system and we can analyze its dynamic behavior. In this section, we first introduce the mathematical models of "human mind" and "learning algorithm".

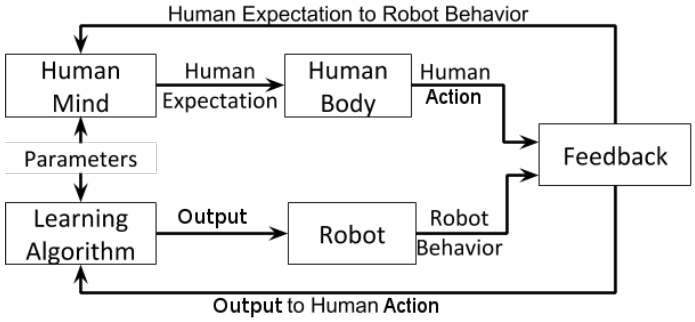

Figure 5.1: Layout for a co-learning system between a human and a learning algorithm

The chatter behavior in the co-learning system with binary actions can be defined as follows:

Definition 5.1.1. *If there exists a t_0 such that $A_t = -A_{t-1}$, $\lambda_t = -\lambda_{t-1}$, and $A_t = -\lambda_t$ that holds for all $t > t_0$, then we say chatter occurs in the co-learning system with binary actions.*

Since we only allow binary values for A_t and λ_t, the chatter behavior in co-learning is a limit cycle in the dynamics. This chatter is not desired for co-learning. Therefore, we will analyze the co-learning system to determine the condition when chatter occurs. And based on this analysis, we show that HAMEA is designed in order to prevent chatter.

5.2 Modeling

5.2.1 Human Learning Model

To model the human mind, we use a well studied psychological model called the Rescorla-Wagner model [43], which has been used to describe human learning that associates a conditioned stimuli with an unconditioned stimuli. We consider the arrival of a robot as the conditioned stimuli and the behavior of the robot as the unconditioned stimuli. Then the Rescorla-Wagner model describes how the human learns to predict the robot's behavior from its arrival.

The Rescorla-Wagner model is described by a difference equation

$$V_t = (1-\gamma)V_{t-1} + \gamma\lambda_{t-1} \tag{5.1}$$

where γ is a constant with real value in $[0, 1]$ representing how much weight the human puts on new information and V_t is a real number in the range of $[-1, 1]$, representing the internal state of the human at the t-th interaction. And we denote V_0 as the initial value of V_t before the human starts interacting with the robot , which is defined by the Rescorla-Wagner model as being in the range of $[-1, 1]$.

Lemma 5.2.1. *Since* $V_0 \in [-1, 1]$ *then* $V_t \in [-1, 1]$ *for all* $t > 0$

Proof. Assume that $V_{t-1} \in [-1, 1]$. Since $-1 \leq V_{t-1} \leq 1$ then $-1(1-\gamma) + \gamma\lambda_{t-1} \leq (1-\gamma)V_{t-1}+\gamma\lambda_{t-1} \leq 1(1-\gamma)+\gamma\lambda_{t-1}$. From equation 5.1 we know that $(1-\gamma)V_{t-1}+\gamma\lambda_{t-1}$, therefore.

$$-1 + \gamma(\lambda_{t-1} + 1) \leq V_t \leq 1 + \gamma(\lambda_{t-1} - 1) \tag{5.2}$$

Since λ_{t-1} is either -1 or 1, the possible values for the minimum of V_t , $-1+\gamma(\lambda_{t-1}+1)$ are -1 or $-1 + 2\gamma$. And the possible values for the maximum of V_t , $1 + \gamma(\lambda_{t-1} - 1)$ are $1 - 2\gamma$ and 1.

Since $\gamma \in [0, 1]$ the possible values for the minimum of V_t are -1 and $[-1, 1]$, and thus the smallest possible value for the minimum of V_t is -1. The possible values for the maximum of V_t are $[-1, 1]$ and $[1]$, and thus the largest possible value for the maximum of V_t is 1. Thus, $V_t \in [-1, 1]$ if $V_{t-} \in [-1, 1]$. Since $V_0 \in [-1, 1]$ then by induction $V_t \in [-1, 1]$, for any $t > 1$. □

The human action is determined by the value of V_t following the rule

$$A_t = \begin{cases} 1 & \text{if } V_t \geq 0 \\ -1 & \text{if } V_t < 0 \end{cases} \tag{5.3}$$

The human subject is only allowed to make two possible choices of the actions represented by the two values of A_t. Correspondingly, our learning algorithm will control the robot to produce two reactions respectively.

5.2.2 Multi Expert Algorithm

For the 'Learning Algorithm' block in figure 5.1, one of the two algorithms we will consider is the MEA. The model of this algorithm is presented in chapter 3. However we will need to make some modifications to the algorithm. The first will be explicitly setting $N = 2$. The second modification will be, in order to more easily integrate with values from the Rescorla-Wagner model, $\lambda = \{1, -1\}$.

Because of the change from the second output label from 2 to -1 the equation for selecting lambda is slightly changed from equation 3.1 in that in the case of a tie the maximum value is chosen instead of the minimum. This is so that output 1 is still chosen in the case of a tie, despite the fact that the second output is valued at -1 instead of 2. The equations 3.2, and 3.3 for the values of R and n remain the same

$$\lambda_t = \min(\text{argmax}\{W_1, W_{-1}\})) \tag{5.4}$$

Algorithm 8 Multi Expert Algorithm

1: Set $W_1 = W_{-1} = 0.5$
2: Choose selection λ_t with equation (5.4)
3: **if** Error ($\lambda_t = -A_t$) **then**
4: $\quad W_{\lambda_t} = \frac{W_{\lambda_t}}{2}$
5: **else if** Success($\lambda_t = A_t$) **then**
6: $\quad$ **if** $W_{\lambda_t} < 0.5$ **then**
7: $\quad\quad W_{\lambda_t} = 2W_{\lambda_t}$
8: $\quad$ **else**
9: $\quad\quad W_{\lambda_t} = W_{\lambda_t}$
10: $\quad$ **end if**
11: **end if**

The three variables $[R, s, \lambda_t,]$ are used to define the automata of MEA as shown in figure 5.2. And based on line 1 of algorithm 8, the initial state is $R_0 = n_0 = \lambda_0 = 1$.

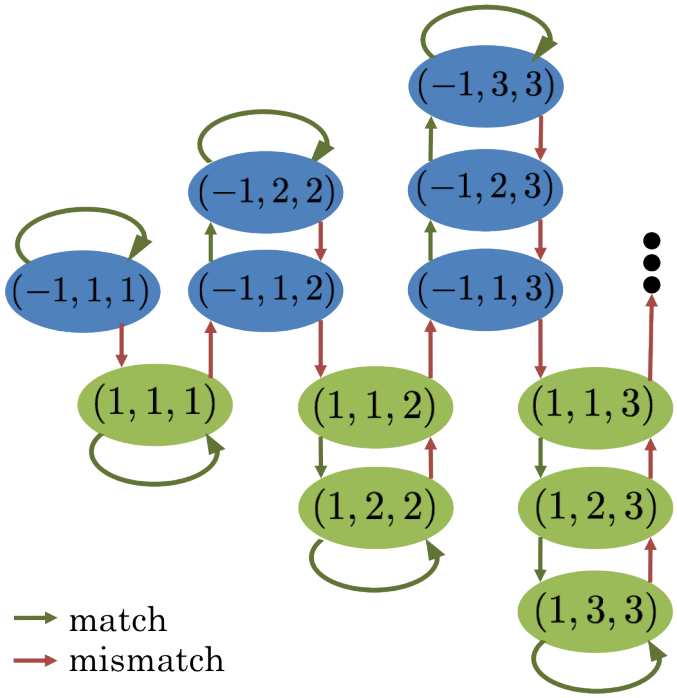

Figure 5.2: MEA Automata with each state defined in terms of $[\lambda, R, n]$

The update equations for each variable are as follows.

$$\lambda_t = \begin{cases} -\lambda_{t-1} & \text{if } \lambda_{t-1} \neq A_t \text{ and } R_{t-1} = 1 \\ \lambda_{t-1} & \text{otherwise} \end{cases} \tag{5.5}$$

$$R_t = \begin{cases} R_{t-1} + 1 & \text{if } \lambda_{t-1} = A_{t-1} \text{ and } R_{t-1} < n_{t-1} \\ R_{t-1} & \text{if } \lambda_{t-1} = A_{t-1} \text{ and } R_{t-1} = n_{t-1} \\ R_{t-1} - 1 & \text{if } \lambda_{t-1} \neq A_{t-1} \text{ and } R_{t-1} > 1 \\ R_{t-1} & \text{if } \lambda_{t-1} \neq A_{t-1} \text{ and } R_{t-1} = 1 \end{cases} \tag{5.6}$$

$$n_t = \begin{cases} n_{t-1} + 1 & \text{if } \lambda_{t-1} = -1 \neq A_t \text{ and } R_{t-1} = 1 \\ n_{t-1} & \text{otherwise} \end{cases} \tag{5.7}$$

We assume that the robot enacts a control law enabling it to preform behavior λ_t, and has detection capabilities sufficient to detect the human action A_t. It is obvious that we cannot directly observe the human's internal parameters γ or $V(t)$, but only observe the human's action A_t. The robot learning algorithm can influence the parameter $V(t)$ but it cannot influence γ_t.

5.2.3 Human Aware Multi Expert Algorithm

In this section, we present HAMEA, which is a revised version of MEA designed to prevent chatter during the co-learning process. The model of this algorithm is presented in chapter 3. However we will need to make the same modifications to this algorithm as we did to algorithm 8 in the previous subsection.

The three variables $[R, s, \lambda_t,]$ are used to define the automata of MEA as shown in figure 5.2. And based on line 1 of algorithm 8, the initial state is $R_0 = n_0 = \lambda_0 = 1$.

This new algorithm produces the same update for λ by equation (5.5), and n by equation

Algorithm 9 Human Aware Multi Expert Algorithm

Set $W_1 = W_{-1} = 0.5$
Choose selection λ_t with equation (5.4)
if Error ($\lambda_t = -A_t$) **then**
    $W_{\lambda_t} = \frac{W_{\lambda_t}}{2}$
    if selection λ_{t+1} from equation (5.4) $\neq \lambda_t$ **then**
        $W_{\lambda_{t+1}} = \begin{cases} 2W_{\lambda_{t+1}} & \text{if } W_{\lambda_{t+1}} < 0.5 \\ 0.5 & \text{otherwise} \end{cases}$
    end if
else if Success($\lambda_t = A_t$) **then**
    if $W_{\lambda_t} < 0.5$ **then**
        $W_{\lambda_t} = 2W_{\lambda_t}$
    else
        $W_{\lambda_t} = W_{\lambda_t}$
    end if
end if

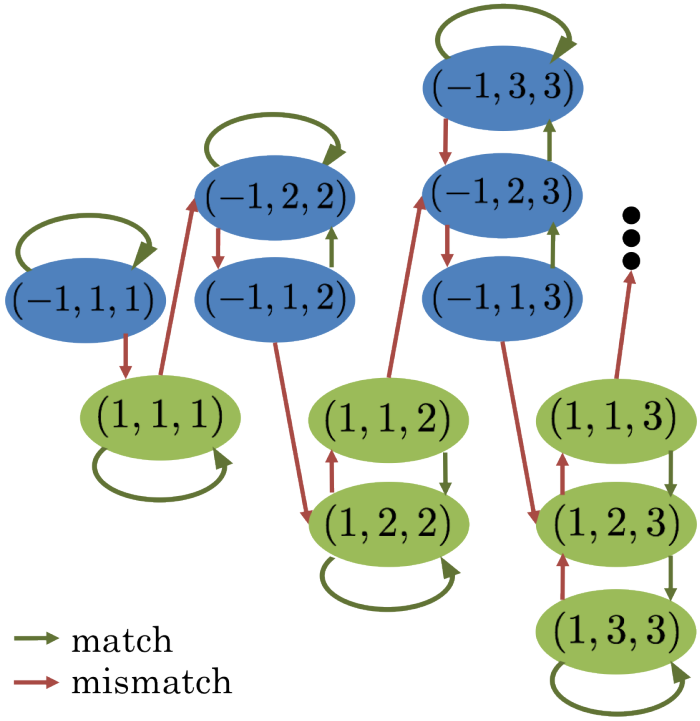

Figure 5.3: HAMEA Automata with each state defined in terms of $[\lambda, R, n]$

(5.7). However the update equation for R has changed as shown in equation (5.8).

$$R_t = \begin{cases} R_{t-1} + 1 & \text{if } \lambda_{t-1} = A_{t-1} \text{ and } R_{t-1} < n_{t-1} \\ R_{t-1} & \text{if } \lambda_{t-1} = A_{t-1} \text{ and } R_{t-1} = n_{t-1} \\ R_{t-1} - 1 & \text{if } \lambda_{t-1} \neq A_{t-1} \text{ and } R_{t-1} > 0 \\ 1 & \text{if } \lambda_{t-1} \neq A_{t-1} \text{ and } R_{t-1} = 0 \text{ and } n_t = 1 \\ 2 & \text{if } \lambda_{t-1} \neq A_{t-1} \text{ and } R_{t-1} = 0 \text{ and } n_t > 1 \end{cases} \tag{5.8}$$

5.3 Chatter Analysis

The human is described by the two internal parameters V_t and γ, where V_t is the state that changes over time. For notation simplicity we drop the index t on A_t and V_t when there is no confusion. Now let $V \in [-1, 1]$ and $A \in \{-1, 1\}$. Let us define the set B_{A_t, r_t} as the set of all values of (γ, V_t) that produce output A_t and takes a minimum of r_t errors from time t to produce the output $-A_t$. This set is important for chatter analysis since any (γ, V_0) starting in this set will generate chatter eventually. We will compute this set in different settings.

Lemma 5.3.1. $B_{-1,1} = \{(V, \gamma) | \gamma \in [0, 1], V \in [-1, 1],$ *and* $V \in [\frac{-1}{1-\gamma} + 1, 0)\}$

Proof. We know that $\gamma \in [0, 1]$ and $V \in [-1, 1]$. And by equation (5.1), $V_t = (1-\gamma)V_{t-1} + \gamma\lambda_{t-1}$. The set produces output $A_t = -1$ and requires 1 error to switch to 1. Therefore, an error must occur at time t, i.e. $\lambda_t = 1$ and A_{t+1} is 1. In order to have $A_{t+1} = 1$, $V_{t+1} \geq 0$ must hold. Then we can find the condition for V_t as follows,

$$0 \leq (1 - \gamma)V_t + \gamma.$$

$$\frac{-\gamma}{1 - \gamma} \leq V_t.$$

And since the action $A_t = -1$, V_t must be less than 0. The range of V_t is then

$$\frac{-1}{1-\gamma} + 1 \leq V_t < 0 \tag{5.9}$$

□

Lemma 5.3.2. *For an arbitrary number of errors* $r > 1$ $B_{-1,r} = \{(V, \gamma) | \gamma \in [0, 1], V \in [-1, 1],$ *and* $V \in [\frac{-1}{(1-\gamma)^r} + 1, \frac{-1}{(1-\gamma)^{r-1}} + 1)\}$

Proof. We know that $\gamma \in [0, 1]$ and $V \in [-1, 1]$. By equation (5.1), $V_t = (1-\gamma)V_{t-1} + \gamma\lambda_{t-1}$. Additionally, let $V_0 = V$ be the initial conditions before any errors occur. And let $V_n \geq 0$ since the output changes only after r errors. Thus $V_i < 0 \ \forall i = 0...n-1$. Additionally $\lambda_0 = ... = \lambda_{r-1} = 1$ Using the same process from Lemma 5.3.1 we can see that V must satisfy the following equations:

$$\frac{-1}{(1-\gamma)} + 1 \leq V_{r-1} < 0$$

$$\frac{-1}{(1-\gamma)^2} + \frac{1-\gamma}{1-\gamma} \leq V_{r-2} < \frac{-1}{(1-\gamma)} + 1$$

$$\frac{-1}{(1-\gamma)^3} + 1 \leq V_{r-3} < \frac{-1}{(1-\gamma)^2} + 1$$

The solutions is then

$$\frac{-1}{(1-\gamma)^r} + 1 \leq V_0 = V < \frac{-1}{(1-\gamma)^{r-1}} + 1 \tag{5.10}$$

Therefore, $B_{-1,r} = \{(V, \gamma) | \gamma \in [0, 1], V \in [-1, 1] V \in [\frac{-1}{(1-\gamma)^r} + 1, \frac{-1}{(1-\gamma)^{r-1}} + 1\}$ □

We can then conclude that the following Lemmas hold by symmetric arguments.

Lemma 5.3.3. $B_{1,1} = \{(V, \gamma) | \gamma \in [0, 1], V \in [-1, 1],$ *and* $V \in [0, \frac{1}{1-\gamma} - 1)\}$

Lemma 5.3.4. $B_{1,r} = \{(V,\gamma)|\gamma \in [0,1], V \in [-1,1],$ *and* $V \in [\frac{1}{(1-\gamma)^{r-1}} - 1, \frac{1}{(1-\gamma)^r} - 1)\}$

After obtaining the representation of the sets, we now show that these sets are not empty. In other words, we want to show that there always exist γ and V that are in these sets.

Lemma 5.3.5. *For any finite* r, $B_{A,r}$ *is not empty.*

Proof. If $A = 1$ or -1, since $\frac{1}{1-\gamma} > 1$ can always be satisfied for some γ, we know that the sets $B_{1,1}$ or $B_{-1,1}$ in Lemmas 3.1, 3.2, 3.3 and 3.4 are not empty.

If $r > 1$ we can look at Lemma 5.3.4 and see that $B_{1,r} = \{(V,\gamma)|\gamma \in [0,1], V \in [-1,1]V \in [\frac{1}{(1-\gamma)^{r-1}} - 1, \frac{1}{(1-\gamma)^{r-1}} - 1\}$. We want to show that there always exist γ so that the set $B_{1,r}$ is not empty. This requires $\frac{1}{(1-\gamma)^{r-1}} - 1 \leq 1$ which becomes $\frac{1}{2} \leq (1-\gamma)^{r-1}$

Note that $\frac{1}{(1-\gamma)^{r-1}} - 1$ is monotonically increasing as r increases. Thus as r goes towards infinity we want to show that there will be (V,γ) within the set. Since $\lim_{r\to\infty} \frac{1}{2}^{\frac{1}{r}} = 1$ which is less than or equal to $1-\gamma$ when $\gamma = 0$ which is within the possible range of γ, then for all finite r there will be a sufficiently small γ so that we can find V to make $B_{1,r}$ not empty. □

Next, we show that one error at time $t-1$ will cause a switching of output at time t for the values of (γ, V_{t-1}) that are in $B_{A_{t-1},1}$.

Lemma 5.3.6. *If an error occurs in the set* $B_{A_{t-1},1}$ *at time* $t-1$ *then at time* t *the state is in set* $B_{-A_{t-1},1}$

Proof. If $A_{t-1} = 1$ then an error means that $\lambda_{t-1} = -1$. We also know from Lemma 5.3.3 that $0 \leq V_{t-1} < \frac{1}{1-\gamma} - 1$. And from equation (5.1) that $V_t = (1-\gamma)V_{t-1} + \gamma\lambda_{t-1}$. Which leads to the following equation.

$$-\gamma \leq (1-\gamma)V_{t-1} - \gamma = V_t < 0 \tag{5.11}$$

Since $\gamma \in [0,1]$ we know that $-\gamma \geq -\frac{1}{1-\gamma} + 1$. This means that V_t is within the set $B_{-A_{t-1},1}$.

If $A_{t-1} = -1$ then an error means that $\lambda_{t-1} = 1$. Using the same analysis as before we get

$$0 \leq (1-\gamma)V_{t-1} + \gamma = V_t < \gamma \tag{5.12}$$

And since $\gamma \in [0,1]$ $\gamma \leq \frac{1}{1-\gamma} - 1$ which means that V_t is within the set $B_{-A_{t-1},1}$. □

5.3.1 MEA Chatter Analysis

We can then give a sufficient condition for chatter for the MEA algorithm as follows:

Lemma 5.3.7. *For the MEA, if $\lambda_{t-1} = -A_{t-1}$ and $R_{t-1} = r_{t-1} = 1$, then chatter will occur for some values of (γ, V_{t-1}).*

Proof. By equation (5.6), at time t $R_t = 1$. And by equation (5.4) $\lambda_t = -\lambda_{t-1}$. As shown by Lemma 5.3.6 if the initial state is within the set $B_{A_{t-1},1}$ and $\lambda_{t-1} = -A_{t-1}$, then $A_t = -A_{t-1}$ and $r_t = 1$.

This means that at time t an error is still produced so that $\lambda_t = -A_t$ and $R_t = r_t = 1$. Thus for all future times an error is still produced. Hence and $\lambda_t = -A_t$ $R_t = r_t = 1$, which will then trigger $A_t = -A_{t-1}$ and $\lambda_t = \lambda_{t-1}$. This leads to chatter according to Definition 5.1.1. □

Therefore, we can conclude that chatter happens for certain initial values.

Theorem 5.3.8. *For MEA a V_0 and γ in set $B_{-1,1}$ will produce chatter.*

Proof. The initial state of MEA is given as $R_0 = s_0 = \lambda_0 = 1$. Thus by Lemma 5.3.7 if γ and V_0 are within set $B_{-1,1}$ the system will produce chatter. □

5.3.2 HAMEA

In order to show that HAMEA does not generate chatter, we show that all possible values of γ and V do not lead to chatter.

Theorem 5.3.9. *If $A_t = \lambda_t$ then there is no chatter after t.*

Proof. If $A_t = \lambda_t$ then based on the update equations (5.1) and (5.3), $A_{t+1} = A_t$. From the update equation (5.4) for λ, $\lambda_{t+1} = \lambda_t$. Thus there will be no chatter after t if $A_t = \lambda_t$. □

Lemma 5.3.10. *For set B_{A_t,r_t}, if $r_t > 1$ and $\lambda_t \neq A_t$ then the set $B_{A_{t+1},r_{t+1}}$ is identical to the set $B_{A_t,-1+r_t}$*

Proof. If $r_t > 1$ and $A_t = -1$ we can use Lemma 5.3.2 which defines the $V_t \in [\frac{-1}{(1-\gamma)^{r_t}} + 1, \frac{-1}{(1-\gamma)^{r_t-1}} + 1)$. Using equation (5.1) we find the range for V_{t+1} after as $[\frac{-1}{(1-\gamma)^{r_t-1}} + 1, \frac{-1}{(1-\gamma)^{r_t-2}} + 1)$ which would be in set $B_{A_t,-1+r_t}$

If $r_t > 1$ and $A_t = 1$ we can use Lemma 5.3.4 which defines the $V_t \in [\frac{1}{(1-\gamma)^{r_t-1}} - 1, \frac{1}{(1-\gamma)^{r_t}} - 1)$. Using equation (5.1) we find the range for V_{t+1} after as $[\frac{1}{(1-\gamma)^{r_t-2}} - 1, \frac{1}{(1-\gamma)^{r_t-1}} - 1)$ which would be in set $B_{A_t,-1+r_t}$.

Thus after an error at time t, if $r > 1$ then $B_{A_{t+1},r_{t+1}} = B_{A_t,-1+r_t}$ □

Theorem 5.3.11. *If $A_t = -\lambda_t$ and $R_t \neq r_t$ then there is no chatter after t.*

Proof. Consider $R_t > r_t$. Since $A_t \neq \lambda_t$ an error occurs at time t. This can lead to two distinct output depending on if $r_t = 1$ or $r_t > 1$.

If $r_t = 1$ then by Lemma 5.3.6, $A_{t+1} = -A_t$. And since $R_t > r_t = 1$, the HAMEA update equations (5.5) and (5.8) give $R_{t+1} = R_t - 1$ and $\lambda_{t+1} = \lambda_t$. Since $\lambda_{t+1} = A_{t+1}$ by Theorem 5.3.9 there is no chatter.

If $r_t > 1$ then by Lemma 5.3.10, $A_{t+1} = A_t$, and $r_{t+1} = r_t - 1$. And since $R_t > r_t > 1$, the HAMEA update equations (5.5) and (5.8) give $R_{t+1} = R_t - 1$ and $\lambda_{t+1} = \lambda_t$. Thus $R_{t+1} = R_t - 1 > r_{t+1} = r_t - 1$ and $A_{t+1} \neq \lambda_{t+1}$. By induction, after a total of $\Delta t = r_t - 1$ iterations, $R_{t+\Delta t} > r_{t+\Delta t} = 1$, which as shown above for $r_t = 1$, produces no chatter.

Now consider $r_t > R_t$. Since $A_t \neq \lambda_t$ an error occurs at time t. This can lead to two distinct output depending on if $R_t = 1$ or $R_t > 1$.

If $R_t = 1$ then by HAMEA update equation (5.5), $\lambda_{t+1} = -\lambda_t$. And since $r_t > R_t = 1$ by Lemma 5.3.10 we know $A_{t+1} = A_t$. Since $\lambda_{t+1} = A_{t+1}$ by Theorem 5.3.9 there is no chatter.

If $R_t > 1$ then by the HAMEA update equations (5.5) and (5.8) give $R_{t+1} = R_t - 1$ and $\lambda_{t+1} = \lambda_t$. And by Lemma 5.3.10, $A_{t+1} = A_t$, and $r_{t+1} = r_t - 1$. And since $r_t > R_t > 1$, Thus $r_{t+1} = r_t - 1 > R_{t+1} = R_t - 1$ and $A_{t+1} \neq \lambda_{t+1}$. By induction, after a total of $\Delta t = r_t - 1$ iterations, $r_{t+\Delta t} > R_{t+\Delta t} = 1$, which as shown above for $r_t = 1$, produces no chatter.

Thus all possible situations that satisfy $A_t = -\lambda_t$ and $R_t \neq r_t$ have been covered. None of these situations lead to chatter. So when $A_t = -\lambda_t$ and $R_t \neq r_t$ then there is no chatter. □

Theorem 5.3.12. *If $A_t \neq \lambda_t$ and $R_t = r_t$ then there is no chatter when using HAMEA.*

Proof. If $A_t \neq \lambda_t$ and $R_t = r_t$ then using HAMEA update equations (5.5) and (5.8), and Lemma 5.3.10, after time $\Delta t = R_t - 1$, $A_{t+\Delta t} = -\lambda_{t+\Delta t}$ and $R_{t+\Delta t} = r_{t+\Delta t} = 1$. By Lemma 5.3.6, $A_{t+\Delta t+1} = -A_{t+\Delta t}$ and $r_{t+\Delta t+1} = 1$.

The following cases must be considered to show that no chatter happens. The first case is if $s_{t+\Delta t+1} > 1$. The second is if $s_{t+\Delta t+1} = 1$.

If $s_{t+\Delta t+1} > 1$ then by HAMEA update equations (5.8) and (5.5), $R_{t+\Delta t+1} = 2$ and $\lambda_{t+\Delta t+1} = -\lambda_{t+\Delta t}$. Thus $R_{t+\Delta t+1} \neq r_{t+\Delta t+1}$ and $\lambda_{t+\Delta t+1} = -A_{t+\Delta t+1}$ which will not produce chatter according to Theorem 5.3.11.

If $s_{t+\Delta t+1} = 1$ then by HAMEA update equations (5.8) and the first when $\lambda_{t+\Delta t+1} = -1$, and the second when $\lambda_{t+\Delta t+1} = 1$.(5.5), $R_{t+\Delta t+1} = 1$ and $\lambda_{t+\Delta t+1} = \lambda_{t+\Delta t}$. This can lead to two cases $\lambda_{t+\Delta t+1} = -1$ and $\lambda_{t+\Delta t+1} = 1$.

If $\lambda_{t+\Delta t+1} = -1$ then by the HAMEA update equations(5.7), (5.6 B), (5.5), $s_{t+\Delta t+2} = 2$, $R_{t+\Delta t+2} = 2$ and $\lambda_{t+\Delta t+2} = -\lambda_{t+\Delta t+1}$ and by Lemma 5.3.6, $A_{t+\Delta t+2} = -A_{t+\Delta t+1}$ and $r_{t+\Delta t+2} = 1$. Thus $R_{t+\Delta t+2} \neq r_{t+\Delta t+2}$ and $\lambda_{t+\Delta t+2} = -A_{t+\Delta t+2}$ which will not produce chatter from Theorem 5.3.11.

If $\lambda_{t+\Delta t+1} = 1$ then by the HAMEA update equations (5.7), (5.6), (5.5), $s_{t+\Delta t+2} = 1$, $R_{t+\Delta t+2} = 1$ and $\lambda_{t+\Delta t+2} = -\lambda_{t+\Delta t+1}$ and by Lemma 5.3.6, $A_{t+\Delta t+2} = -A_{t+\Delta t+1}$ and

$r_{t+\Delta t+2} = 1$. This is now the same as the previous case, $\lambda_{t+\Delta t+1} = -1$, and so this case also does not produce chatter.

Since all the cases considered do not produce chatter, and they cover all possible cases. If $A_t \neq \lambda_t$ and $R_t = r_t$ then there is no chatter when using HAMEA. □

Combining Theorem 5.3.9, Theorem 5.3.11, and Theorem 5.3.12 we conclude that there is no values of γ and V_t that can produce chatter for the HAMEA.

5.4 Simulation

To support the chatter analysis for MEA and HAMEA simulations were run for both algorithms. Each algorithm was started at $R_0 = 1$,$n_0 = 1$, and $\lambda_0 = 1$. Each algorithm was run using the Rescorla-Wagner model to model the human they were interacting with. Values of γ and V_0 of the Rescorla-Wagner model were iterated through, and the amount of steps that it took the algorithm to reach a steady state was recorded for each γ and V_0 pair. If 50 iterations passed without a steady state being reached the algorithm was considered to chatter.

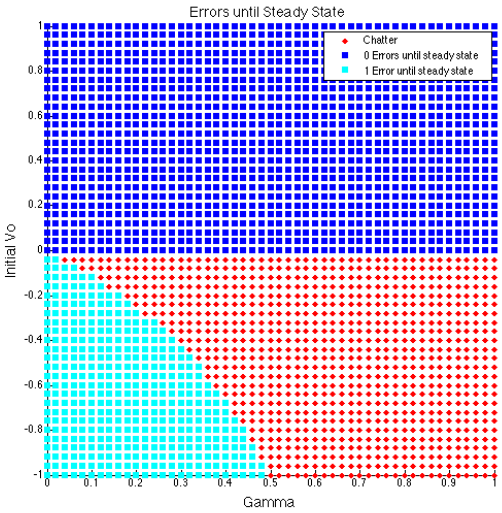

Figure 5.4: Simulation results for steps until steady state using MEA based off of γ and V_0

The red area in Figure 5.4 for MEA indicates the values of γ and V_0 where chatter

occurs. These V_0 and γ are within the set $B_{-1,1}$, which our analysis has predicted. As comparison, Figure 5.5 shows the steps required to reach steady state when HAMEA is used. There is no chatter present e.g. no red region. The values in $B_{-1,1}$ only requiring 3 steps to reach steady state, which matches with our analysis.

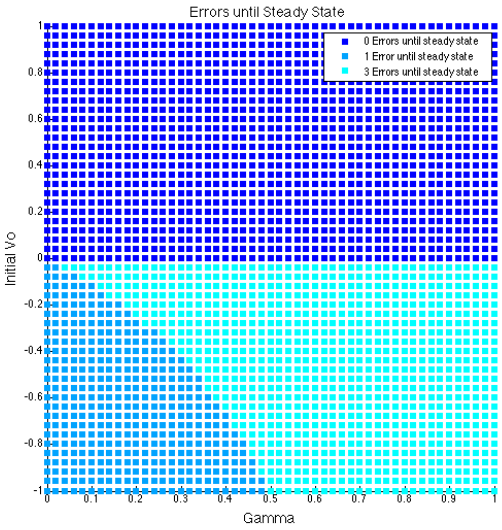

Figure 5.5: Simulation results for steps until steady state using HAMEA based off of γ and V_0

5.5 Implementation

HAMEA was implemented onto the the GT-MAB. The algorithm was able to take input from the blimp and return an action selection that the blimp was able to execute. It was then able to take in feedback and HAMEA updates the concept learnt by GT-MAB. The update rule executed in an average of 8.87×10^{-3} seconds over 6 Trials. This shows that HAMEA can be implemented without a noticeable increase in the computation time compared to MEA. It is well situated to perform future interaction studies.

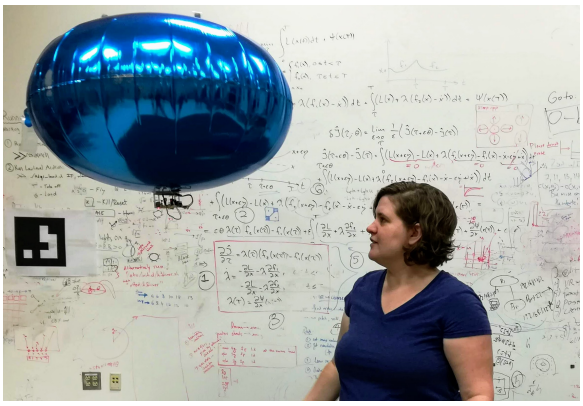

Figure 5.6: Human with GT-MAB running HAMEA.

CHAPTER 6
CONCLUSION

This thesis gives a novel approach for modeling simple N-Expert online ensemble learning algorithms as automata. Four online ensemble learning algorithms: the WMA, the Winnow algorithm, the MEA, and the HAMEA, were modeled as automata using this method. This modeling allows for a more comprehensive understanding of the performance of the online ensemble learning algorithm, the environment the algorithm interacts with and the behavior of the learning algorithm when interacting with a second learning system.

The automata models were combined with a probability distribution to allow for the performance of the learning algorithms to be modeled with Markov chains. Performance metrics such as the consistency and adaptiveness of an algorithm can be calculated using the mean hitting times of these Markov chains. Since these performance metrics are important for the implementation of online ensemble learning algorithms into applications such as human robot interaction, this computation aids in the potential for these algorithms to be utilized and designed for these types of applications.

These automata models were also combined with a second learning system in order to identify and prevent undesirable behaviors. One such undesirable behavior is chatter, which occurs when a learning algorithm continually changes its prediction, without reaching a constant prediction of the second learning system's intention. Modeling the second learning system as a human by using the Rescorla-Wagner model we were able to identify the situations where chatter will occur and to identify aspects of the design, such as shown in HAMEA, that prevent chatter.

Future work consists of expanding the automata creation and Markov chain analysis to the cases where the number of possible outputs is less than the number of experts used. It also consists of expanding the parameter identification and chatter analysis to cases where

$n > 2$. As well as the development and implementation of online ensemble learning algorithms for real world problems with strict performance constraints.

Appendices

APPENDIX A

EXPERIMENTAL RESULTS

Table A.1: Subject 1 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 1	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1,1	1	1,1
1	1	1	1	2	1	1,1	1	1,1
1	1	1	1	3	1	1,1	1	1,1
1	1	1	1	4	1	1,1	1	1,1
1	1	1	1	5	1	1,1	1	1,1
2	1	1	1	6	1	1,1	1	1,1
1	2	1	1	5	2	1,1	2	1,1
1	1	1	1	6	1	1,2	1	2,2
1	1	1	1	7	1	2,2	1	2,2
1	1	1	1	8	1	2,2	1	2,2
2	1	1	1	9	1	2,2	1	2,2
1	2	1	1	8	1	1,2	1	1,2
1	1	1	1	9	1	2,2	1	2,2
	1	1	1	10	1	2,2	1	2,2
Error	4		2		3		3	
n	3		1		2		2	

Table A.2: Subject 2 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 2	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1,1	1	1,1
1	1	1	1	2	1	1,1	1	1,1
1	1	1	1	3	1	1,1	1	1,1
1	1	1	1	4	1	1,1	1	1,1
1	1	1	1	5	1	1,1	1	1,1
2	1	1	1	6	1	1,1	1	1,1
2	2	1	1	5	2	1,1	2	1,1
2	2	1	1	4	2	1,1	2	1,1
2	2	1	1	3	2	1,1	2	1,1
2	2	1	1	2	2	1,1	2	1,1
2	2	1	1	1	2	1,1	2	1,1
2	2	1	2	1	2	1,1	2	1,1
	2	1	2	2	2	1,1	2	1,1
Error	1		6		1		1	
n	1		1		1		1	

Table A.3: Subject 3 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 3	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1,1	1	1,1
1	1	1	1	2	1	1,1	1	1,1
2	1	1	1	3	1	1,1	1	1,1
1	2	1	1	2	2	1,1	2	1,1
1	1	1	1	3	1	1,2	1	2,2
1	1	1	1	4	1	2,2	1	2,2
1	1	1	1	5	1	2,2	1	2,2
1	1	1	1	6	1	2,2	1	2,2
2	1	1	1	7	1	2,2	1	2,2
1	2	1	1	6	1	1,2	1	1,2
1	1	1	1	7	1	2,2	1	2,2
1	1	1	1	8	1	2,2	1	2,2
	1	1	1	9	1	2,2	1	2,2
Error	4		2		3		3	
n	3		1		2		2	

Table A.4: Subject 4 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 4	λ	R	λ	R	λ	R, n	λ	R, n
2	1	1	1	1	1	1,1	1	1,1
2	2	1	2	1	2	1,1	2	1,1
2	2	1	2	2	2	1,1	2	1,1
1	2	1	2	3	2	1,1	2	1,1
2	1	1	2	2	1	1,2	1	2,2
1	2	1	2	3	2	1,2	1	1,2
1	1	1	2	2	1	1,3	1	2,2
2	1	1	2	1	1	2,3	1	2,2
2	2	1	2	2	1	1,3	1	1,2
2	2	1	2	3	2	1,3	2	2,2
2	2	1	2	4	2	2,3	2	2,2
1	2	1	2	5	2	3,3	2	2,2
	1	1	2	4	2	2,3	2	1,2
Error	6		5		7		6	
n	4		1		3		2	

Table A.5: Subject 5 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 5	λ	R	λ	R	λ	R, n	λ	R, n
2	1	1	1	1	1	1 , 1	1	1 , 1
1	2	1	2	1	2	1 , 1	2	1 , 1
1	1	1	1	1	1	1 , 2	1	2 , 2
1	1	1	1	2	1	2 , 2	1	2 , 2
2	1	1	1	3	1	2 , 2	1	2 , 2
2	2	1	1	2	1	1 , 2	1	1 , 2
1	2	1	1	1	2	1 , 2	2	2 , 2
2	1	1	1	2	1	1 , 3	2	1 , 2
2	2	1	1	1	2	1 , 3	2	2 , 2
1	2	1	2	1	2	2 , 3	2	2 , 2
1	1	1	1	1	2	1 , 3	2	1 , 2
1	1	1	1	2	1	1 , 4	1	2 , 3
2	1	1	1	3	1	2 , 4	1	3 , 3
2	2	1	1	2	1	1 , 4	1	2 , 3
	2	1	1	1	2	1 , 4	1	1 , 3
Error	7		9		10		9	
n	4		3		4		3	

Table A.6: Subject 6 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 6	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1 , 1	1	1 , 1
1	1	1	1	2	1	1 , 1	1	1 , 1
2	1	1	1	3	1	1 , 1	1	1 , 1
1	2	1	1	2	2	1 , 1	2	1 , 1
2	1	1	1	3	1	1 , 2	1	2 , 2
1	2	1	1	2	2	1 , 2	1	1 , 2
2	1	1	1	3	1	1 , 3	1	2 , 2
1	2	1	1	2	2	1 , 3	1	1 , 2
2	1	1	1	3	1	1 , 4	1	2 , 2
1	2	1	1	2	2	1 , 4	1	1 , 2
2	1	1	1	3	1	1 , 5	1	2 , 2
1	2	1	1	2	2	1 , 5	1	1 , 2
2	1	1	1	3	1	1 , 6	1	2 , 2
	2	1	1	2	2	1 , 6	1	1 , 2
Error	11		6		11		7	
n	6		1		6		2	

Table A.7: Subject 7 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 7	λ	R	λ	R	λ	R, n	λ	R, n
2	1	1	1	1	1	1 , 1	1	1 , 1
1	2	1	2	1	2	1 , 1	2	1 , 1
2	1	1	1	1	1	1 , 2	1	2 , 2
1	2	1	2	1	2	1 , 2	1	1 , 2
2	1	1	1	1	1	1 , 3	1	2 , 2
1	2	1	2	1	2	1 , 3	1	1 , 2
1	1	1	1	1	1	1 , 4	1	2 , 2
1	1	1	1	2	1	2 , 4	1	2 , 2
2	1	1	1	3	1	3 , 4	1	2 , 2
1	2	1	1	2	1	2 , 4	1	1 , 2
1	1	1	1	3	1	3 , 4	1	2 , 2
1	1	1	1	4	1	4 , 4	1	2 , 2
1	1	1	1	5	1	4 , 4	1	2 , 2
1	1	1	1	6	1	4 , 4	1	2 , 2
	1	1	1	7	1	4 , 4	1	2 , 2
Error	8		7		7		5	
n	5		4		4		2	

Table A.8: Subject 8 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 8	λ	R	λ	R	λ	R, n	λ	R, n
2	1	1	1	1	1	1 , 1	1	1 , 1
2	2	1	2	1	2	1 , 1	2	1 , 1
2	2	1	2	2	2	1 , 1	2	1 , 1
1	2	1	2	3	2	1 , 1	2	1 , 1
1	1	1	2	2	1	1 , 2	1	2 , 2
2	1	1	2	1	1	2 , 2	1	2 , 2
2	2	1	2	2	1	1 , 2	1	1 , 2
1	2	1	2	3	2	1 , 2	2	2 , 2
2	1	1	2	2	1	1 , 3	2	1 , 2
2	2	1	2	3	2	1 , 3	2	2 , 2
1	2	1	2	4	2	2 , 3	2	2 , 2
2	1	1	2	3	2	1 , 3	2	1 , 2
	2	1	2	4	2	2 , 3	2	2 , 2
Error	7		5		7		6	
n	4		1		3		2	

Table A.9: Subject 9 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 9	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1 , 1	1	1 , 1
2	1	1	1	2	1	1 , 1	1	1 , 1
1	2	1	1	1	2	1 , 1	2	1 , 1
2	1	1	1	2	1	1 , 2	1	2 , 2
1	2	1	1	1	2	1 , 2	1	1 , 2
2	1	1	1	2	1	1 , 3	1	2 , 2
1	2	1	1	1	2	1 , 3	1	1 , 2
2	1	1	1	2	1	1 , 4	1	2 , 2
1	2	1	1	1	2	1 , 4	1	1 , 2
2	1	1	1	2	1	1 , 5	1	2 , 2
1	2	1	1	1	2	1 , 5	1	1 , 2
2	1	1	1	2	1	1 , 6	1	2 , 2
	2	1	1	1	2	1 , 6	1	1 , 2
Error	11		6		11		7	
n	6		1		6		2	

Table A.10: Subject 10 Experimental Results

	WMA		Winnow		MEA		HAMEA	
Subject 10	λ	R	λ	R	λ	R, n	λ	R, n
1	1	1	1	1	1	1 , 1	1	1 , 1
2	1	1	1	2	1	1 , 1	1	1 , 1
2	2	1	1	1	2	1 , 1	2	1 , 1
1	2	1	2	1	2	1 , 1	2	1 , 1
1	1	1	1	1	1	1 , 2	1	2 , 2
2	1	1	1	2	1	2 , 2	1	2 , 2
2	2	1	1	1	1	1 , 2	1	1 , 2
1	2	1	2	1	2	1 , 2	2	2 , 2
1	1	1	1	1	1	1 , 3	2	1 , 2
2	1	1	1	2	1	2 , 3	1	2 , 3
1	2	1	1	1	1	1 , 3	1	1 , 3
2	1	1	1	2	1	2 , 3	1	2 , 3
1	2	1	1	1	1	1 , 3	1	1 , 3
2	1	1	1	2	1	2 , 3	1	2 , 3
	2	1	1	1	1	1 , 3	1	1 , 3
Error	9		9		8		9	
n	5		3		3		3	

APPENDIX B

PARAMETER IDENTIFICATION

B.1 Full State

In the Full State (FS) analysis the entire state history is available as well as the complete transition matrix. From the full state history a transition matrix can be calculated from equation B.3.

$$\mathbf{P_{i,j}} = \mathbb{P}(\mathbf{Z_1} = \mathbf{j} | \mathbf{Z_0} = \mathbf{i}) \tag{B.1}$$

This transition matrix can then be compared with the algorithm's identified transition matrix $\mathbf{T}$, which for the two expert case is given in terms of p_1 and p_2 where $p_2 = 1 - p_1$. This comparison is done by means of the variable $s_{i,j}$ which is 1 when $T_{i,j}$ is given as p_1 and 0 otherwise.

$$s_{i,j} = \begin{cases} 1 & \text{if } \mathbf{T}_{i,j} = p_1 \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

This comparison allows for the computation of an averaged value of p_1 from $\mathbf{P}_{i,j}$, $s_{i,j}$, and k_i which is the number of times that state i was visited as shown in equation B.3.

$$p_1 = \sum_{i=1} \sum_{j=1} \frac{\mathbf{P}_{i,j} s_{i,j}}{k_i} \tag{B.3}$$

Lemma B.1.1. *When $N = 2$ the full state calculation produces the same result for p_1 as calculating p_1 directly from the record of expected outputs.*

Proof. Calculating p_1 directly from the record of expected outputs can be found by computing $\frac{C_1}{c}$ where C_1 is the number of steps where output 1 was expected, and c is the total

number of steps.

Each time output 1 is expected, if the algorithm chose output 1 then a success occurred and the state changed using the transition labeled as p_1. If the algorithm chose output $\lambda \neq 1$ then an error occurred and the state changed using the transition labeled as $1 - p_\lambda$. Since $N = 2$ then there are only two expected outputs and $p_1 + p_2 = 0$ so that $1 - p_2 = p_1$. This same analysis holds for each time output 2 is expected as well. Thus each time output one is expected corresponds to one time that the algorithm changed from state z_0 to state z_1 along the transition labeled p_1.

Thus the transition probability labeled p_1 transition from state z is calculated by $\frac{C_{1,z}}{c_z}$. Where $C_{1,z}$ is the number of times that output 1 was expected when the algorithm was in state z and c_z is the number of times that the algorithm was in state z.

This probability is then normalized by $\frac{c_z}{c}$ to become $\frac{C_{1,z}}{c}$. When this fraction is summed over all possible states that gives $\frac{C_1}{c}$, which is equivalent to calculating p_1 directly from the record of expected outputs □

B.2 First State Last State

In the First State Last State (FSLS) analysis there are three key pieces of information from the run-time of the algorithm. The first is the initial state of the algorithm. The second is the final state of the algorithm. And the third is the total number of states, k. FSLS analysis also requires a full Markov chain and transition matrix with only one variable. When $N = 2$ this can be done by making p_1 the variable and $p_2 = 1 - p_1$. The goal of this analysis is to determine p_1 which can be done utilizing a max likelihood estimation.

$$\hat{p_1}_{\text{MLE}} = \text{argmax}_{p_1} \mathbf{u}_0^T \mathbf{T}^k(p_1)\mathbf{u}_k \tag{B.4}$$

Equation B.4 describes the equation used to determine p_1. $\mathbf{u}$ is the observed likelihood distribution from the initial state $t = 0$ and the final state $t = k$. $\mathbf{T}$ is the transition matrix

that was calculated from the learning algorithm.

REFERENCES

[1] L. Kuncheva, "Classifier ensembles for changing environments," in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, F. Roli, J. Kittler, and T. Windeatt, Eds., vol. 3077, Springer Berlin Heidelberg, 2004, pp. 1–15, ISBN: 978-3-540-22144-9.

[2] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*, C. Zhang and Y. Ma, Eds., Springer US, 2012, pp. 1–34, ISBN: 978-1-4419-9325-0.

[3] M. A. Wiering and H. Van Hasselt, "Ensemble algorithms in reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 930–936, 2008.

[4] M. Oudah, V. Babushkin, T. Chenlinangjia, and J. W. Crandall, "Learning to interact with a human partner," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2015, pp. 311–318.

[5] M. Zambelli and Y. Demiris, "Online multimodal ensemble learning using self-learned sensorimotor representations," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 2, pp. 113–126, 2017.

[6] D. Parikh and R. Polikar, "An ensemble-based incremental learning approach to data fusion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 437–450, 2007.

[7] Y. Xu, X. Cao, and H. Qiao, "An efficient tree classifier ensemble-based approach for pedestrian detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 41, no. 1, pp. 107–117, 2011.

[8] S Kotsiantis, K. Patriarcheas, and M Xenos, "A combinational incremental ensemble of classifiers as a technique for predicting students performance in distance education," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 529–535, 2010.

[9] H. S. Mohammed, J. Leander, M. Marbach, and R. Polikar, "Comparison of ensemble techniques for incremental learning of new concept classes under hostile non-stationary environments," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, IEEE, vol. 6, 2006, pp. 4838–4844.

[10] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," in *Foundations of Computer Science, 1989., 30th Annual Symposium on*, IEEE, 1989, pp. 256–261.

[11] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Machine learning*, vol. 2, no. 4, pp. 285–318, 1988.

[12] C. Mesterharm, "A multi-class linear learning algorithm related to winnow," in *Advances in Neural Information Processing Systems*, 2000, pp. 519–525.

[13] C. Mesterharm and D. F. Hsu, "Combinatorial fusion with on-line learning algorithms," in *2008 11th International Conference on Information Fusion*, IEEE, 2008, pp. 1–8.

[14] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[15] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. 8, pp. 2755–2790, 2007.

[16] K. Crammer, Y. Mansour, E. Even-Dar, and J. W. Vaughan, "Regret minimization with concept drift.," in *COLT*, 2010, pp. 168–180.

[17] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters*, vol. 33, no. 2, pp. 191–198, 2012.

[18] S. G. Soares and R. Araújo, "An on-line weighted ensemble of regressor models to handle concept drifts," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 392–406, 2015.

[19] B. Krawczyk and A. Cano, "Online ensemble learning with abstaining classifiers for drifting and noisy data streams," *Applied Soft Computing*, vol. 68, pp. 677–692, 2018.

[20] P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust ensemble learning for mining noisy data streams," *Decision Support Systems*, vol. 50, no. 2, pp. 469–479, 2011.

[21] V. Babushkin, M. Oudah, T. Chenlinangjia, A. Alshaer, and J. W. Crandall, "On-line learning in repeated human-robot interactions," in *2014 AAAI Fall Symposium Series*, 2014.

[22] J. W. Crandall, "Non-myopic learning in repeated stochastic games," *ArXiv preprint arXiv:1409.8498*, 2014.

[23] C. Young, A. Khan, and F. Zhang, "Adaptiveness and consistency of expert based learning algorithms selecting reactions to human movements," in *American Control Conference (ACC), 2017*, IEEE, 2017, pp. 1530–1535.

[24] S. Nikolaidis, J. Forlizzi, D. Hsu, J. Shah, and S. Srinivasa, "Mathematical models of adaptation in human-robot collaboration," *ArXiv preprint arXiv:1707.02586*, 2017.

[25] S. Nikolaidis, D. Hsu, and S. Srinivasa, "Human-robot mutual adaptation in collaborative tasks: Models and experiments," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 618–634, 2017.

[26] M. Barlier, J. Perolat, R. Laroche, and O. Pietquin, "Human-machine dialogue as a stochastic game," in *16th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL 2015)*, 2015.

[27] D. Helbing, P. Molnár, I. J. Farkas, and K. Bolay, "Self-organizing pedestrian movement," *Environment and planning B: Planning and design*, vol. 28, no. 3, pp. 361–383, 2001.

[28] T. Nomura, T. Kanda, T. Suzuki, and K. Kato, "Prediction of human behavior in human–robot interaction using psychological scales for anxiety and negative attitudes toward robots," *IEEE transactions on robotics*, vol. 24, no. 2, pp. 442–451, 2008.

[29] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro, "Interactive robots as social partners and peer tutors for children: A field trial," *Human-computer interaction*, vol. 19, no. 1, pp. 61–84, 2004.

[30] N. Yao, E. Anaya, Q. Tao, S. Cho, H. Zheng, and F. Zhang, "Monocular vision-based human following on miniature robotic blimp," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 3244–3249.

[31] Q. Tao, J. Cha, M. Hou, and F. Zhang, "Parameter identification of blimp dynamics through swinging motion," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 2018, pp. 1186–1191.

[32] N.-s. Yao, Q.-y. Tao, W.-y. Liu, Z. Liu, Y. Tian, P.-y. Wang, T. Li, and F. Zhang, "Autonomous flying blimp interaction with human in an indoor space," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 1, pp. 45–59, 2019.

[33] A. Blum, "Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain," *Machine Learning*, vol. 26, no. 1, pp. 5–23, 1997.

[34] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Design of an office-guide robot for social interaction studies," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 4965–4970.

[35] ——, "Evaluation of passing distance for social robots," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication*, IEEE, 2006, pp. 315–320.

[36] M Lauckner, F Kobiela, and D Manzey, "??hey robot, please step back!??-exploration of a spatial threshold of comfort for human-mechanoid spatial interaction in a hallway scenario," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, IEEE, 2014, pp. 780–787.

[37] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch, "Social and collaborative aspects of interaction with a service robot," *Robotics and Autonomous Systems*, vol. 42, no. 3, pp. 223–234, 2003.

[38] S.-Y. Chung and H.-P. Huang, "A mobile robot that understands pedestrian spatial behaviors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2010, pp. 5861–5866.

[39] ——, "Predictive navigation by understanding human motion patterns," *International Journal of Advanced Robotic Systems*, vol. 8, no. 1, pp. 52–64, 2011.

[40] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[41] N. Privault, "First step analysis," in *Understanding Markov Chains*, Springer, 2013, pp. 95–116.

[42] C. Young and F. Zhang, "A learning algorithm to select consistent reactions to human movements," in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 6152–6157.

[43] R. A. Rescorla, "A theory of pavlovian conditioning: The effectiveness of reinforcement and non-reinforcement," *Classical conditioning II: Current research and theory*, 1972.

VITA

Carol Young was born in San Antonio, Texas. She spent her early life in Grafenwoehr, Germany, Washington, D.C. and Savannah, Georgia. She then attended Rensselaer Polytechnic Institute and graduated with a B.S. in Aeronautical and Mechanical Engineering. Following graduation she worked at Electric Boat, before returning to school to pursue her passion in Robotic Engineering.